ANTEPRIMA DELPHI 2005

RIVOLUZIONARIO! PROGRAMMARE IN .NET CON OBJECT PASCAL

VERSIONE PLUS RIVISTA+LIBRO+CD €9,90

VERSIONE STANDARD RIVISTA+CD €6,90

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • FEBBRAIO 2005 • ANNO IX, N.2 (88)

ioProgrammo Anno IX - N°2 (88)

XML e Web Services WSE2 • Il tuo primo GPS • OpenGIS & MySQL • Applicazioni autoaggiornanti

EDIZIONI MASTER

XML E WEB SERVICES WSE2

SPAM KILLER

Un sistema intelligente che riconosce la posta indesiderata, la elimina e impara aggiornandosi con le segnalazioni degli utenti

- Gestire l'autenticazione degli utenti
- Bloccare la posta indesiderata lato client
- Aggiornare il server tramite comunicazioni criptate

COSTRUISCI E PROGRAMMA IL TUO PRIMO GPS

- ✔ Un progetto alla portata di tutti per realizzare un rilevatore di posizione
- ✔ Usa PHP, MySQL e le estensioni OpenGis per tracciare la rotta

.NET

APPLICAZIONI AUTOAGGIORNANTI

Funzionalità come Windows Update nel tuo software e usando il tuo sito web

A CACCIA DEL BACO NASCOSTO!

La guida definitiva al Tracing & Debugging di un'applicazione .NET

Imparare a programmare AL VIA I NUOVI CORSI

- •ASP.NET Apprendere le tecnologie Microsoft per lo sviluppo di applicazioni Web
- •MACROMEDIA FLASH ACTIONSCRIPT 2.0 Sviluppare ad oggetti con il più potente linguaggio orientato al multimedia

IOPROGRAMMO WEB

ASP.NET

Al riparo dagli attacchi degli Hacker! Difendiamoci dal SQL Injection

IL NUOVO PHP 5

Impara a realizzare codice migliore con il modello a oggetti

JAVA

CREARE REPORT AVANZATI

Come realizzare grafici, statistiche e presentazioni con il linguaggio di SUN

MASSIMA SICUREZZA

SSL e certificati X509, per rendere impenetrabili le vostre comunicazioni

C++

VIDEOGAMES

Simulazione della realtà con animazioni e movimento in ambienti 3D

9 771128 594009 50088

EDIZIONI MASTER www.edmaster.it

VISUAL BASIC PRELEVARE DATI E GRAFICI DA UN FOGLIO EXCEL

Anno IX - N.ro 2 (88) - Febbraio 2005 - Periodicità Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
MarCom Director: Luca Perfetti
Marketing Manager: Antonio Meduri
Editor: Gianfranco Forlino
Coordinamento redazionale: Raffaele del Monaco
Redazione: Fabio Farnesi
Collaboratori: M. Autiero, L. Barbieri, M. Bigatti, D. Bochicchio, L. Buono, F. Cozzolino, T. D'Argenio, C. De Sio Cesari, F. Grimaldi, F. C. Ferracchiati, M. Locuratolo, A. Marroccelli, F. Mestrone, G. Natili, P. Perrotta, L. Spuntoni, I. Venuti, F. Vaccaro.
Segreteria di Redazione Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale: SET S.r.l.
Coordinamento Tecnico: Piero Mannelli
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo Basic (11 numeri) €52,90 sconto 30% sul prezzo di copertina di €75,90 ioProgrammo con Libro (11 numeri + libro) €76,00 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 31/03/05
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Ariberto, 24 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master Servizio Clienti - Via Ariberto, 24 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
tel.02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelliera, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore - zona ASI Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Gennaio 2005

Edizioni Master edita: Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, TV e Satellite, Win Extra, Home entertainment, Digital Japan, Digital Music, Horror Mania, Computer Bild, ioProgrammo Extra, Le Collection.

Questo mese su ioProgrammo

ioProgrammo enciclopedico

Prima di ogni cosa desidero farvi i miei migliori auguri per un sereno 2005. Passato il periodo di festa, il normale ciclo lavorativo rientra pigramente nei suoi ritmi consueti. La maggior parte di voi sarà già di nuovo immerso nei propri progetti. Il 2005 dal punto di vista strettamente informatico si apre con l'annuncio di un nuovo boom del settore internet. Come sempre, agli annunci previsionali, noi programmatori aspettiamo che facciano seguito fatti ben precisi. Tuttavia se le previsioni fossero confermate dovremmo aspettarci di vedere arrivare sulle nostre scrivanie molti progetti relativi ad applicazioni che girano su Internet.
ioProgrammo con la cautela di sempre non si sbilancia totalmente in questa direzione, ritenendo che la programmazione standalone sia ancora un pezzo importante del mercato italiano del Software, tuttavia avrete notato che abbiamo iniziato ad offrirvi un supporto più costante in questa direzione. D'altra parte è importante avere una conoscenza di insieme di svariate tecniche oltre che una conoscenza precisa del proprio settore per orientarsi nel complicato mondo dello sviluppo. Abbandonata la cautela, desidero invece raccontarvi di una piccola esperienza consumata cercando la parola ioProgrammo su Google NewsGroup. Con mio sommo piacere ho trovato messaggi di ogni tipo con date variabili da 1996 a 2005, molto spesso ho trovato riferimenti ad ioProgrammo in risposta a problemi di programmazione, altre volte ho trovato citato nella ricerca bibliografica in relazione a tesi di laurea, spesso e volentieri ho trovato riferimenti all'uso dei nostri articoli come base per lo sviluppo di applicazioni aziendali, infine ho trovato aste di vecchi numeri di ioProgrammo e qualcuno che desiderava acquistare la collezione completa dal numero 1 al numero 88. Mi sono scoperto a contemplare la mia collezione di ioProgrammo dal numero 1 all'88 bellamente disposta e ordinata nel ripiano più accessibile della mia libreria, io stesso la consulto spessissimo per trovare soluzioni a questo o a quel problema e mi sono ritrovato a pensare al valore enciclopedico del nostro Magazine.
Chi segue da sempre questa rivista possiede ora una collezione dal valore tecnico/commerciale fuori del comune, assolutamente degno delle migliori enciclopedie!

Fabio Farnesi

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.
Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **iop437** *Password:* **sk25122004**

XML e Web Services WSE2
SPAM KILLER

Un sistema intelligente che riconosce la posta indesiderata, la elimina e impara aggiornandosi con le segnalazioni degli utenti

- Gestire l'autenticazione degli utenti
- Bloccare la posta indesiderata lato client
- Aggiornare il server tramite comunicazioni criptate

pag. 14

COSTRUISCI E PROGRAMMA IL TUO PRIMO GPS

http://forum.ioprogrammo.it

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

RUBRICHE

ioProgrammo cerca articolisti freelance competenti nei seguenti argomenti:

Javascript, Python, Perl, ASP.NET, PHP, Flash, Security

Inviare curriculum dettagliato a ioProgrammo@edmaster.it

RIVISTA + CD-ROM in edicola

I contenuti del CD-Rom

INTERNET

ASP.NET Maker 1.1 pag. 121

Make codeworks or edit texts locally or directly on a remote server.

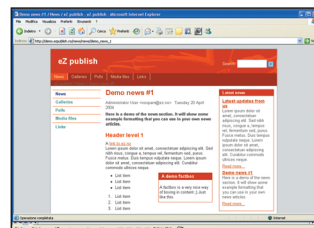
aspnetmkr.exe

Apache 1.3.33/2.0.52 pag. 115

Uno dei server Web più usati al mondo

Directory /Apache

EZPublish 3.1.4 pag. 115

Un sistema di sviluppo per CMS

Diretory /ezpublish

PHP ADS New 2.0 pag. 115

Gestisce in modo profesionale le campagne banner su Internet

Directory /PHPAdsNew

PHPMyAdmin 2.6.0

Molto probabilmente il Frontend più usato al mondo per MySQL

Directory /PHPMyAdmin

NucleusCMS pag. 115

Probabilmente il primo CMS orientato al blogging

Directory /Nucleus

PHPWiki pag. 115

Ottimo per la creazione di documentazione scritta in modo collaborativo

Directory /PHPWiki

STRUMENTI

Quick License Manager 2.1 pag. 121

Proteggi il tuo software dai pirati con

Prodotti del mese

Delphi 2005

Borland apre a .NET

Ne parliamo diffusamente in questo stesso numero di ioProgrammo. Si tratta della nuova piattaforma di programmazione di Borland che, fra le novità più interessanti, aggiunge il supporto a .NET. In parole povere è possibile sviluppare con Delphi, ovvero il primo ambiente RAD ad avere conquistato il mercato, utilizzando Object Pascal per la piattaforma .NET. Chiaramente non si tratta dell'unica novità di questo ambiente. Ma creare la mia prima applicazione ASP.NET in Object Pascal con Delphi è stato emozionante come fu circa 10 anni fa creare la mia prima applicazione con un ambiente RAD quale era allora Delphi 1.0.

[pag.110]

SwishLite

Crea in 5 Minuti un'applicazione FLASH

SwhishLite è un grande prodotto. Dedicato ai programmatori e ai grafici, specializzati in campi diversi da quelli dell'animazione vettoriale, non vogliono comunque perdere la possibilità di rendere esteticamente accattivanti le proprie applicazioni. Consente di realizzare effetti sorprendenti, offre un tool minimo ma molto bilanciato di funzionalità per interagire e personalizzare i filmati. Esiste, anche la possibilità di importare filmati realizzati con Flash per una successiva elaborazione. Insomma si tratta di un prodotto completo che può aiutarvi in tutte quelle situazioni dove la vostra vena artistica non è sufficiente a supportare le vostre capacità di programmatore.

[pag.116]

Stylus Studio 6.0

Completo! per la gestione di documenti XML

Un editor XML tra i più potenti e completi che possiate trovare. L'interfaccia è ricchissima di pulsanti e opzioni ma, grazie ad una organizzazione esemplare, riesce a non disorientare già dal primo utilizzo. La finestra di editing è organizzata a tab e consente dunque di passare rapidamente da un file all'altro; inoltre, per ogni file XML, è possibile scegliere fra quattro diverse visualizzazioni: Text, Tree, Grid e Schema.
Ancora più interessanti le opzioni di visualizzazione disponibili per trasformazioni XSLT.
Anche verso i database Stylus Studio dispone di utili funzioni sia per l'import sia per l'export.

[pag.119]

Poseidon for UML Community Edition 3.0

Creare diagrammi UML e fare il reverse engineering del codice Java

Poseidon è un software per la progettazione UML di livello professionale. La versione Community Edition che qui presentiamo è gratuita e consente sia la generazione di diagrammi UML sia il reverse engineering a partire da codice sorgente Java. Implementato completamente in Java, può girare su qualsiasi piattaforma. I diagrammi possono essere esportati in svariati formati (gif, ps, eps e svg), pieno supporto per il drag & drop, interessanti funzionalità per il reverse engineering di sorgenti Java.

[pag.120]

RIVISTA + LIBRO + CD-ROM in edicola

I contenuti del libro

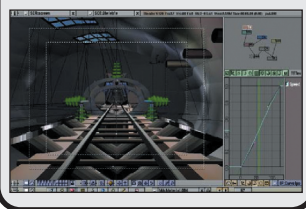
C# REFERENCE

C# e .NET rappresentano molto probabilmente l'innovazione più importante dal punto di vista dello sviluppo introdotta da Microsoft nel corso degli ultimi anni.
Se da un lato si può ritenere che Visual Basic e C++ siano ancora molto amati dai programmatori, è anche vero che la migrazione verso C# procede a ritmo incessante. Le novità introdotte nel linguaggio e nel compilatore lo rendono estremamente innovativo rispetto ai precedenti e tale che il suo utilizzo conduce a realizzare applicazioni sempre più vicine all'evoluzione del mercato. Il libro C# reference è un manuale di rapida consultazione che contiene i dettagli dei costrutti più importanti del linguaggio. Si tratta di un reference dedicato a chi ha iniziato da poco a programmare e non ha ancora la completa padronanza del linguaggio, ma anche di un riferimento completo per coloro che pur conoscendo da qualche tempo C# desiderano avere un punto di riferimento rapido per la risoluzione di piccoli problemi di sintassi o di utilizzo dei costrutti. Al solito si tratta di un reference da tenere di fianco al monitor pronto per essere consultato nel momento del dubbio.

Prova subito

Blender

L'Editor 3D per realizzare complessi ambienti anche per la programmazione di videogiochi

File su CD: blender.zip

Librerie e strumenti

ASP, C#, Java, Perl, VB.NET, Python, PHP, C++, XML pag. 115

questo interessante tool
qlm.exe

DotNet2UML 2.1 pag. 121
Legge gli assembly prodotti da .NET e crea una rappresentazione UML portabile
AssemblyMetadata2XMI.zip

XML:Wrench 1.2.1 pag. 122
Manipolazione di file XML
xmlwrench-v121.exe

Notepad++ 2.6 pag. 122
Un super notepad!
npp.2.6.Installer.exe

JetBrains ReSharper 1.0.4 pag. 122
Un assistente per C# in Visual Studio
ReSharper1.0.4.exe

ToolbarCreator 1.0 pag. 122
Crea la tua toolbar per Internet Explorer
ToolbarCreator10Setup.exe

JFrameBuilder 3.0.1 pag. 121
Per creare sophisticate interface Java
JFB_301.zip

Synopsis 1.1.5
Costruisci intere applicazioni solo con il drag&drop
synopsis_demo_setup.exe

ColorCache 3.0 pag. 121
Un aiuto nella scelta dei colori
cche3000.exe

SuperEdi 3.5 pag. 121
Un editor piccolo e funzionale per gli sviluppatori
SuperEdi-3.5.U.exe

PureBasic 3.92 pop pag. 120
BASIC... a tempo di record!
PureBasic_Demo.exe

Jcreator 3.1.0 pag. 117
Un grande editor per Java
Directory /Jcreator

PHPEclipse pag. 117
Un plugin per sviluppare applicazioni PHP con eclipse
Directory /PHPEclipse

MySQL Control Center pag. 117
L'interfaccia grafica per gestire DB Mysql
Directory /Mysql/mysqlcc-0.9.4-win32.zip

MySQL Connector .NET pag. 117
Il connector per usare MySQL direttamente da .NET
/Mysql/mysql-connector-net-1.0.2-gamma.zip

Tomcat 50 pag. 123
L'application server per le vostre applicazioni JSP.
Directory /tomcat

LINGUAGGI

PHP 4.3.9/5.0.2 pag. 117
Il linguaggio di scripting per il web
Directory /PHP

Python 2.3.4 pag. 118
Un linguaggio orientato agli oggetti con tanto di supporto a classi ed ereditarietà
Directory: /Python

DATABASE

Softsilver Transformer 2.5.2 pag. 120
Riversare dati da ODBC in XML
st25setup.msi

MySQL 4.1.7 pag. 118
Il server di database OpenSource più diffuso al mondo
Directory /Mysql

News

SLACKWARE DI NUOVO IN PISTA

Si era temuto, nei mesi passati, per un problema fisico di Patrick Volkerding attuale maintainer e creatore della distribuzione Linux Slackware.
Un annuncio sul sito ufficiale aveva fatto credere che proprio per problemi di salute il Team Leader di Slackware avrebbe dovuto dare forfait.
L'annuncio aveva gettato nello scompiglio milioni di utilizzatori della distribuzione. Di fatto Linux Slackware è una delle distribuzioni più amate dagli utilizzatori del sistema operativo del Pinguino. L'abbandono di Patrick Volkerding senza dubbio avrebbe causato qualche problema di mantenimento degli aggiornamenti e upgrade a nuove versioni.
È recente invece l'annuncio che Patrick Volkerding sta meglio, è rientrato al suo posto di lavoro e continuerà come sempre a mantenere le redini del Team di Slackware. Un ben tornato anche da parte nostra! Patrick Volkerding è uno dei programmatori che ha svolto un lavoro realmente significativo per l'intera comunità mondiale nel corso degli anni.

http://www.slackware .com

RFID: CRESCITA VERTIGINOSA GRAZIE A WAL MART

Il business dell'RFID sta crescendo in fretta grazie alle pressioni di Wal Mart, la più grande catena di supermercati in america. Wal Mart aveva fissato gennaio 2005 come deadline per l'adozione degli RFID da parte di tutti i suoi fornitori: giganti del calibro di Gillette, Kraft e Procter & Gamble hanno dovuto piegarsi al dictat, investendo complessivamente oltre 250 milioni di dollari in tag e attrezzature.

DURA VITA PER GLI HACKER

Nove anni di carcere sono stati inflitti a un uomo del Michigan per essersi introdotto insieme ad altre due persone nei sistemi informatici del negozio Online della Lowe's Hardware. Si tratta della sentenza più dura che sia stata inflitta negli stati uniti per un crimine informatico, minore di quella inflitta a Kevin Mitchnick l'hacker più famoso della storia. Altri due uomini restano in attesa della sentenza. Uno dei due Adam Timmins è accusato di WarDriving, ovvero di utilizzare un apparato mobile collegato ad un'antenna e di muoversi lungo le strade alla ricerca di connessioni Wireless non protette e dunque attaccabili.

JAVA 6.0 NOME IN CODICE MUSTANG

Sun ha iniziato lo sviluppo di Java 6.0. Il nome in codice del nuovo progetto è Mustang. Le snapshot sono già disponibili per il download all'ndirizzo https://j2se.dev.java.net. Allo stesso indirizzo è possibile partecipare al CVS, ai forum di discussione che possono incidere sulle feature che Sun intende includere nella nuova versione del linguaggio. È interessante notare che J2SE 6.0 sarà rilasciato sotto licenza JRL, ovvero la nuova licenza di SUN nata per favorire l'adozione del linguaggio in ambienti di ricerca o di studio.

IBM DICE ADDIO AI PC

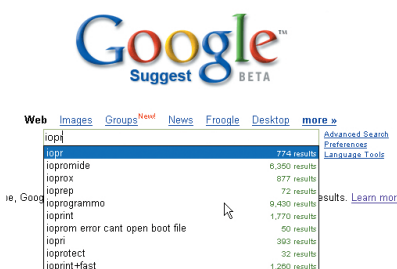
Ormai è ufficiale! Big Blue, il colosso propulsore della moderna informatica, ha abbandonato il settore dei PC. La divisione Personal Computer di IBM è stata venduta alla cinese Lenovo per circa 1.75 miliardi di dollari. In realtà la Lenovo avrebbe acquisito la divisione pc di IBM pagando una parte del debito in azioni, perciò IBM sarebbe ora proprietaria del 18% della Lenovo. A seguito di questa operazione la nuova classifica dei maggiori costruttori di PC al mondo si compone come segue: Dell, HP, Lenovo.

È stata proprio HP la prima società a reagire all'ascesa della Lenovo, immettendo sul mercato cinese un PC dal costo di appena 370 Euro. Si tratta di un AMD Pavillon su cui è precaricato un sistema FreeDos OpenSource. Se queste sono le premesse, c'è da scommettere che la prepotente ascesa di una compagnia "Low Cost" come la Lenovo, supportata dall'acquisizione dell'expertise di IBM, non mancherà di scuotere ulteriormente il già enormente variegato mondo dell'hardware.

TUTTI CONTRO GOOGLE

Sembra proprio che dopo il rilascio della Google Desktop Search Bar si sia scatenato un putiferio fra i rivali del motore di ricerca più famoso al mondo. Così dopo che il rilascio di MSN Search da parte di Microsoft, segue adesso il rilascio della Yahoo Search-Bar. Il prodotto di Yahoo si basa su una versione rivista, migliorata e nativa per Windows del vecchio Lotus Magellan.
La software House incaricata di realizzare il software è la X1 che fa capo alla Idealab di Bill Gross, ideatore e sviluppatore proprio di Lotus Magellan. La nuova Search Bar di Yahoo sarebbe sensibilmente più veloce rispetto alla rivale di Google, non girerebbe in un browser e offrirebbe funzionalità di ricerca avanzate.
Il principale svantaggio sarebbe invece che essendo nativa per Windows il porting verso altri sistemi operativi sembra essere piuttosto complesso.
Il leader in questo settore rimane ancora comunque "Copernic" che non ha ancora annunciato novità di rilievo per quanto riguarda il proprio software.

GOOGLE IL SUGGERITORE

Ci risiamo: ancora una volta, Google esaudisce un nostro sogno prima ancora che l'avessimo sognato! Andate al link che trovate in fondo alla news. Cominciate a scrivere (lentamente) la parola che volete cercare, Google tenterà di indovinare quello che state cercando e vi proporrà una serie di alternative in un elenco a discesa. Un'occhiata alla immagine qui sotto è più esplicativa di mille parole. Il servizio, denominato Google Suggest, è ancora in fase sperimentale e, per il momento, è orientato agli utenti anglofoni. Comunque non mancate di sperimentarlo perché può rivelarsi davvero prezioso: oltre a ridurre il lavoro di digitazione dei più pigri, il servizio consente di esplorare una serie di alternative alla ricerca che avevamo in mente, migliorando e ampliando i risultati che possiamo ottenere. La funzionalità è ottenuta attraverso l'indicizzazione dei miliardi di query processate da Google, indicizzazione che potete apprezzare anche attraverso la pagine zeitgeist proposta da Google, con la classifica delle ricerche più gettonate. Un ultima nota che interesserà i lettori di questa rivista: lo sviluppatore di questo servizio afferma nel suo blog di aver ideato e realizzato la versione beta di Google Suggest in meno di due mesi, e praticamente da solo… diavolo di un Google!

MICROSOFT ACQUISISCE GIANT ANTISPYWARE

Che Microsoft abbia in parte basato i suoi successi acquisendo piccole compagnie dalle potenzialità portentose è un fatto ormai noto. Questa volta è il turno di Giant AntiSpyware, una delle software house che più si è distinta negli ultimi tempi per la realizzazione di software innovativo. La prima beta di un software basato su Giant AntiSpyware ma targato Microsoft è atteso per la fine di Gennaio 2005, non ci sono ancora dettagli sulla politica commerciale che contraddistinguerà la distribuzione di questo tool, eccetto che sarà disponibile per le versioni di Windows a partire da Windows 2000 fino alle più recenti. Parallelamente viene riportato che Microsoft non cesserà di supportate SpamInspector e PopupInspector.

SOBER-I È IL VIRUS PIÙ DIFFUSO DEL MOMENTO

Sophos, una delle aziende leader nel settore della protezione antispam e antivirus, ha reso noto il rapporto sui 10 virus più diffusi del momento. Sarebbe Sober-I, comparso appena nel mese di Novembre ad avere avuto la maggiore diffusione, con oltre il 20% delle segnalazioni, a seguire compaiono Bagle-AU e Netsky-P, questi ultimi due se pure risalgono al mese di Marzo 2004, sarebbero ancora i più fastidiosi per l'utenza finale.
La classifica aggiornata è la seguente

1.	W32/Netsky-P	24,2%
2.	W32/Sober-I	20,4%
3.	W32/Zafi-B	17,9%
4.	W32/Bagle-AU	11,0%
5.	W32/Netsky-D	4,4%
6.	W32/Netsky-Z	3,9%
7.	W32/Bagle-AA	2,6%
8.	W32/Netsky-B	2,4%
9.	W32/MyDoom-O	2,2%
10.	W32/Netsky-Q	2,1%
	Altri	8,9%

Sober-I si diffonde attraverso le e-mail e si nasconde dietro messaggi di testo e oggetti che spesso riguardano proprio la sicurezza, confondendo non poco gli utenti, che spesso incorrono nell'errore del doppio click sull'allegato senza approfondirne la provenienza.
I dati e le classifiche aggiornate della diffusione dei virus possono essere reperiti presso
http://www.sophos.com/virusinfo/infofeed

IN ARRIVO JNBRIDGEPRO 2.2

JNBridge ha annunciato il rilascio di JNBridge PRO 2.2. il tool che consente una stretta interoperabilità fra Java e .NET. In sostanza JNBridge consente di invocare metodi appartenenti a classi Java da ambienti .NET e viceversa.
Garantendo una stretta interoperabilità fra due strumenti che da soli occupano di per sé una straordinaria fetta del mercato dedicato allo sviluppo del software. La compatibilità fra i due ambienti è alta. In questa nuova versione vengono aggiunti il supporto alle funzioni di CallBack nei progetti Java-To.Net, il passaggio di oggetti per valore e la conversione .fra NET's System.DateTime e java.util .Date, e fra .NET System.Decimal e java.math.BigInteger e java. math.BigDecimal. JNBridge sta conquistando lentamente molte quote di mercato, affrontando il problema della convergenza che al momento sembra essere uno dei più sentiti nel mondo della programmazione.

INBox
L'esperto risponde...

Installare IIS in Windows XP Home Edition

Sono molto attratto dalla programmazione .NET, tuttavia mi rendo conto che i costi per me sono eccessivi. Inoltre per provare le pagine in locale dovrei installare IIS, tuttavia io ho Windows XP Home Edition, perciò non sono in grado di installare IIS. A questo punto mi chiedo perché Microsoft fa delle tecnologie così interessanti per un pubblico di pochi eletti. In Italia ci sono circa 1.000.000 di domini ormai registrati, non credo che tutta questa roba sia fatta dalle aziende.

Moulinex 81

Gentile amico, intanto è vero! Stiamo per arrivare al milione di domini italiani registrati WOW! Quanti di questi siano realmente siti attivi è difficile stabilirlo. In ogni caso una buona fetta è sviluppata in PHP, qualcosa in meno in ASP, a ruota seguono ASP.NET e JSP. È altrettanto vero che l'offerta dei provider è variata sensibilmente nell'ultimo anno. Di fatto il mercato che prima prevedeva solo offerte a basso costo Unix + PHP + MySQL, fornisce anche ora soluzioni a basso costo per Windows2003 + ASP .NET + SQL Server, questo può voler solo dire che c'è un nutrito numero di sviluppatori .NET che non utilizza gli stumenti avanzati di Microsoft ma che pure programma tranquillamente in .NET. Il modo migliore per iniziare a sviluppare in ASP.NET senza spendere cifre da capogiro è scaricare Web Matrix da *http://www.asp.net/Default .aspx?tabindex=0&tabid=1*. Si tratta di un tool "ufficiale" di Microsoft che supporta pienamente lo sviluppo ASP.NET anche se abbraccia un modello di licenza non commerciale. Per provare le pagine in locale è infine disponibile "cassini" *http://www.asp.net/Default .aspx?tabindex=6& tabid=41*, un web server dalle dimensioni ridottissime che però vi consente tranquillamente di eseguire i vostri test senza troppi problemi. Esiste anche una soluzione "non ufficiale" ovvero installare "Mono", un progetto OpenSource che adotta le stesse identiche specifiche di Microsoft.NET.
Mono vi mette a disposizione tutto quello che vi serve per programmare in .NET, compreso un editor *Mono-Develop*, e un web Server XSP. Il prezzo da pagare è qualche leggera differenza con il framework di Microsoft, Mono non implementa ancora tutti gli oggetti conosciuti, tuttavia rappresenta una valida alternativa. Inoltre poichè Mono è compatibile sia con Linux che con Windows potrete realizzare progetti multipiattaforma .NET esattamente come accade ormai da molti anni per Java. Infine grazie a un apposito modulo è possibile usare Mono.NET direttamente da Apache.

Gif animate in Visual Basic

**Cari amici di ioProgrammo. Sviluppando in VB6 mi sono trovato di fronte al problema di visualizzare una GIF animata in un controllo di tipo Picture. Devo dire che ho avuto scarso successo, non riesco a visualizzare nessun tipo di Gif animata in quel controllo.
Come posso fare?**

Maurizio

Gentile lettore, effettivamente il controllo PictureBox standard di VB6 non supporta le GIF animate. Un metodo per aggirare il problema è utilizzare un controllo di tipo WebBroser all'interno del quale nascondere la Gif. Ad esempio il controllo in questione potrebbe avere al suo interno il segente codice:

```
<html>
 <body>
  <img src="filename.gif" border="0" />
 </body>
</html>
```

A questo punto per eliminare le barre di scorrimento che sicuramente sporcheranno la visualizzazione dell'immagine è opportuno associare all'evento onLoad del body uno stile che nasconda le barre di scorrimento.

Editor alternativo a Truespace

Salve a tutti, mi voglio complimentare con Alfredo Marroccelli e il resto di ioProgrammo per il lavoro splendido che svolgete. Ogni mese aspetto con impazienza l'uscita della rivista, anche se avrei preferito più argomentazioni sul C++ e le sue enormi risorse. Sono rimasto attratto dall'articolo sull'engine 3D Irrlicht, ma non riesco a caricare file .x creati con Caligari Truespace 3.2, mentre non ho alcun problema con il file "colonna.x" presente nel CD. Uso VC++.Net e vorrei sapere se esiste un modellatore 3D free, in alternativa a Truespace anche se quest'ultimo se pur datato è sempre ottimo.

Tequila74

Buongiorno, un editor OpenSource alternativo a Truespace, se pure con qualche leggera limitazione è Blender. In questo numero di ioProgrammo nel cd allegato alleghiamo la versione 2.35.
Oltre alla potenza intrinseca del prodotto è interessante notare che tutti gli script che ne estendono le funzionalità sono scritti in Python. Inoltre Blender è ben documentato ed esiste un sito italiano con una community abbastanza attiva *http://www .blender.it*

Parametri liberi in C#

Gentile Redazione, prima di porvi la mia domanda, non

**posso mancare di ringraziarvi per il vostro eccezionale lavoro.
Venendo a noi: potreste dirmi se è possibile dichiarare in C# un metodo che accetta un numero variabili di argomenti? E se sì: come?**

Mirko Trovato

Ciao Mirko, grazie per i complimenti. Riguardo alla tua domanda, in C# puoi utilizzare la parola chiave params nella lista dei parametri accettati da un metodo. Quando un parametro è dichiarato params il metodo accetta qualsiasi numero di argomenti:

```
public static void UseParams(params int[] list)
{
    for ( int i = 0 ; i < list.Length ; i++ )
        Console.WriteLine(list[i]);
    Console.WriteLine();
}
```

Se non si vuole limitare i parametri ad un singolo tipo, di può utilizzare la parola chiave object:

```
public static void UseParams2(
                        params object[] list)
{
    for ( int i = 0 ; i < list.Length ; i++ )
        Console.WriteLine((object)list[i]);
    Console.WriteLine();
}
```

Inoltre, ricordati che non sono consentiti altri parametri dopo quello dichiarato come params, e che è consentito l'utilizzo di una sola parola chiave params per ogni dichiarazione di metodo. Ecco come invocare i due metodi che ti ho appena descritto:

```
public static void Main()
{
    UseParams(1, 2, 3);
    UseParams2(1, 'a', "test");

    int[] myarray = new int[3] {10,11,12};
    UseParams(myarray);
}
```

La rosa dei venti in VB.NET

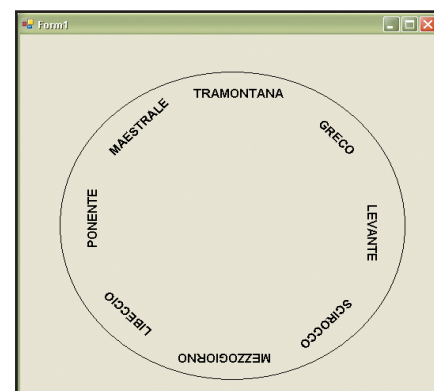
Sto scrivendo una piccola applicazione in VB.NET e ho bisogno di scrivere del testo all'interno di un cerchio. Il testo dovrebbe essere orientato con un angolo diverso per ogni parola… un po' come avviene per la rosa dei venti! Come posso fare?

Gregorio Rosanio

Caro Gregorio, di seguito trovi del codice che fa esattamente quello che chiedi. Come vedi, per ogni scritta abbiamo impostato un angolo diverso (incrementato di 45 gradi per ognuna) sfruttando la classe Matrix del namespace Drawing2D.

```
Public Enum Direction As Integer

    TRAMONTANA  =  0
    GRECO       =  1
    LEVANTE     =  2
    SCIROCCO    =  3
    MEZZOGIORNO =  4
    LIBECCIO    =  5
    PONENTE     =  6
    MAESTRALE   =  7

End Enum
Protected Overrides Sub OnPaint(ByVal e As
        System.Windows.Forms.PaintEventArgs)
    e.Graphics.Clear(Me.BackColor)
    Dim bounds As Rectangle
    Dim g As Graphics
    Dim rotation As Single = 0
    g = e.Graphics
    bounds = New Rectangle(50, 50,
Me.Width
                        - 100, Me.Height - 100)
    Dim rect As System.Drawing.RectangleF
    g.DrawEllipse(Pens.Black, bounds)
    Dim myMatrix As Drawing2D.Matrix
    Dim sf As New StringFormat(
                    StringFormatFlags.NoWrap)
    sf.Alignment = StringAlignment.Center
    myMatrix = g.Transform()
    rect = New System.Drawing.RectangleF(
            bounds.X + 20, bounds.Y + 20,
        bounds.Width - 20, bounds.Height - 20)
    For i As Integer = 0 To 7
        If i > 0 Then
            myMatrix.RotateAt(45, New PointF(
                Me.Width / 2, Me.Height / 2),
            Drawing.Drawing2D.MatrixOrder.Append)

            g.Transform = myMatrix
        End If
        Dim directionString As String
        directionString = System.Enum.GetName(
                        GetType(Direction), i)
        g.DrawString(directionString, New
Font(
                "Arial", 12, FontStyle.Bold),
                    Brushes.Black, rect, sf)
    Next
End Sub
```

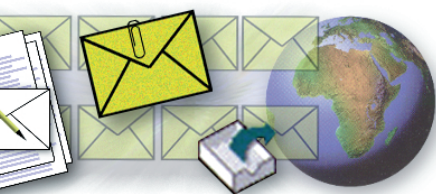
Per testare il codice, è sufficiente generare un nuovo progetto VB.NET in Vi-

sual Studio e copiarlo così com'è.
Il risultato dovrebbe essere simile a quello che vedi in figura.

Documenti XML validi

Cara Redazione, potreste spiegarmi la differenza fra un documento XML valido e uno Well Formed?

Antonio Palermo

Le specifiche XML impongono che tutti i documenti XML siano "well-formed", mentre non è necessario che siano anche validi. Un documento well-formed deve rispettare tutte le caratteristiche specifiche per XML, come ad esempio: evitare tag non correttamente innestati, tutti i riferimenti a informazioni esterne devono essere esplicitate, e così via.
Un documento XML valido, oltre ad essere well-formed, deve rispettare una specifica grammatica indicata in un documento DTD (Document Type Definition). Un DTD (che è a sua volta un documento XML) descrive una struttura per documenti XML.

PER CONTATTARCI
e-mail: ioprogrammo@edmaster.it
Posta: Edizioni Master,
Via Ariberto, 24 - 20123 Milano

NEWSGROUP
Le informazioni nella Rete

ioProgrammo seleziona per voi le discussioni più interessanti dai newsgroup tecnici di programmazione e dal forum di Io Programmo http.//forum.ioprogrammo.it

Visual Basic
Forum di ioProgrammo Visual Basic

EXPORT FILE TXT

Ciao a tutti! Vorrei sapere se è possibile esportare in un file di testo il contenuto di una textbox.
L'applicazione che ho progettato è collegata ad un db. Ho collegato le textbox ad alcuni campi di una tabella access (vers. 2000; mi chiedevo se oltre al fatto di eseguire un bel update sul db in questione, fosse possibile tramite un tasto o funzione esportare gli stessi dati direttamente su di un file di testo.
Grazie a tutti per l'attenzione ;)
belva

Risponde ciufolo

Dopo averti creato il file .txt dove appoggiare i tuoi dati, all'evento clik del pulsante, inserisci il codice:

```
Open App.Path & "\Test.txt" For Append As #3
Print #3, Chr(10)
Print #3, "Testo:"
Print #3, Me.Text1.Text
```

Io il mio file .txt l'ho chiamato Test, tu chiamalo come vuoi ti spiego le righe di codice:

1) Con l'istruzione open apri il file .txt dove salvare ituoi dati.

2) Chr(10) significa la dimensione del carattere

3) Testo: è una riga che ti riporta la scritta testo sul foglio .txt

4) me.text1.text Riporta il contenuto della textbox.

Saluti

XML... perché?

Per motivi di lavoro stavo avvicinandomi ad XML. Ho scaricato progetti di esempio, alcuni tutorial... ma onestamente non ho capito perché io, programmatore, dovrei scegliere di usare XML.
Mi spiego meglio:

ho bisogno di immagazzinare grandi quantità di dati, di effettuare ricerche in modo semplice etc.?!
Ok: scelgo un database.

Ho bisogno di memorizzare i settaggi di una mia applicazione? Scelgo un file ini.

Ora, perché dovrei scegliere XML?
In cosa può migliorare gli strumenti che ho a disposizione?
L'accesso è veloce?
La sintassi semplice?
Le query performanti?
Non ha limiti di grandezza?
Altro?!
Scusate il post un po' da idiota... Ma preferisco cercare di capire, piuttosto che imparare uno strumento a caso...
Hyde

Risponde Wodka40°

Nessuna di tutte quelle ragioni! Un file xml non è né più e né meno che un file di testo. Con tutti i pregi e difetti... XML è un "core" standard di interscambio dati. Assieme ad un file XML puoi fornire uno schema per interpretare i tuoi dati.
Il bello è che chi ha quello schema indipendentemente dalla macchina interpreterà i dati in maniera corretta!
Esiste un MAthXML che di mestiere non fa che interpretare complicate formule matematiche.Tu fornisci un file di testo e in base a quello schema il PC ti ricostruisce la formula. Esiste un ChemXML... ecc. Anche una pagina web puoi descriverla con XML. In .NET è diventato lo standard per appoggiare dati fra una applicazione e l'altra. In VB6 da mdac 2.5, mi sembra di ricordare, hai con il metodo SAVE la possibilità di salvare un recordset anche in XML.
QUINDI: si campa anche senza! Se proietti la tua applicazione nel futuro e la vuoi "aprire" al mondo... prevedi di esportare i tuoi dati anche in formato XML!
P.s. XML funziona bene anche su altri SO... Linux ad esempio... proprio perchè non dipende da un SO! È una soluzione quindi all'interscambio dati fra macchine diverse!

Java
Forum di ioProgrammo - Java

Inversione di stringhe

Ciao e grazie a chi vorrà rispondere. In java 1.5 c'è una proprietà delle stringhe che ne permette l'inversione completa?
Se non fossi stato chiaro, intendo far diventare "ciao" "oaic".
Sono nuovo del linguaggio quindi vi prego d'essere clementi nelle risposte.
ShadowofTruth

Risponde tjqblackdragon

Una possibile soluzione, presente sin dal JDK 1.0.2, è utilizzare l'oggetto StringBuffer che possiede il meto-

do reverse.
Eccoti un possibile esempio:

```
public class InvertiStringa
{
  public static void main(String[] args)
  {
    StringBuffer prova = new
                         StringBuffer("ciao");
    prova.reverse();
    System.out.println(prova); //viene
                               visualizzato oaic
  }
}
```

Installazione di una servlet

Salve ragazzi, chiedo scusa anticipatamente se la domanda che sto per porvi è già stata fatta 1000 volte, ma non mi sono rimesso a leggere tutti i vecchi topic. Io ho una Servlet Java che riesco a lanciare tramite jbuilder. Il problema è che ora vorrei slegarla dall'ambiente di sviluppo e lanciarla ad esempio tramite un link dal desktop.
Il problema è che non ho idea di che comando usare o di che eseguibile lanciare. Dovrei aver già installato Tomcat con jBuilder (anche perchè quando lancio la servlet cn quest'ultimo usa appunto Tomcat), ma io cosa devo scrivere nella riga di comando per lanciare il processo server? Mi sareste d grande aiuto se poteste aiutarmi, o comunque consigliarmi un tutorial o qualcosa di simile.
Vi ringrazio anticipatamente.

fedetallo

Risponde alieno

Ciao, allora vediamo se ti posso aiutare! Di solito per montare una servlet all'interno di un application server (Tomcat nel tuo caso), devi stare attento a un po' di cosette: ammettiamo che l'applicazione che gira sull'application server in cui tu vuoi installare la servlet si chiami "MiaApplicazione".
Quest'ultima avrà uno spazio riservato sul file system sotto tomcat (all'interno di \webapps per essere precisi), nella fattispecie una cartella con il suo nome, quindi "MiaApplicazione".
All'interno troveremo la solita organizzazione, cioè le cartelle \classes e \WEB-INF. all'interno di \WEB-INF è presente il descrittore dell'applicazione "web .xml" all'interno del quale, tra le altre cose, devi inserire le mappature delle servlet che l'applicazione utilizza.
Per ogni servler quello che devi fare è inserire informazioni sulla servlet e la sua mappatura, come segue:

```
<servlet>
<servlet-name>MiaServlet</servlet-name>
<display-name>MiaServlet</display-name>
<servlet-class>path.MiaServlet<
                                /servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>MiaServlet</servlet-name>
<url-pattern>/MiaServlet</url-pattern>
</servlet-mapping>
```

Dove c'è scritto path, devi inserire il path a partire dalla ContextRoot che hai indicato all'interno del descrittore del server "server.xml".
A questo punto, se da una pagina HTML, che dichiara una form al suo interno con attributo action=MiaServlet, nel momento in cui operi la submit della form la servlet che viene richiamata è MiaServlet!! Spero di essere stato abbastanza chiaro e soprattutto di aver capito il tuo reale problema!!
Fammi sapere come te la cavi!!
Buon lavoro.

C#

Forum di ioProgrammo - C#

C# e i documenti di Word

Ciao a tutte/i, qualcuno può darmi qualche dritta per poter aprire (da una form in C#) un file di Word e visualizzarne il contenuto? ciao e grazie...

blue74

Risponde amdbook

Puoi provare in questo modo:

```
System.Diagnostics.Process pr=
           new System.Diagnostics.Process();
pr.StartInfo.FileName ="pathfileword";
pr.Start ();
```

Python

Forum di ioProgrammo - Python

Python è per tutte le Piattaforme?

Ciao, avrei una curiosità: Python può funzionare su tutte le piattaforme? Sia Windows che Linux? Grazie!

lowernautilus

Risponde Salvatore Meschini

Python è disponibile per tutte le piattaforme più comuni: Windows, Unix/Linux, Macintosh. Estistono porting anche per sistemi più o meno "esoterici": PlayStation2, Amiga, Solaris, AS/400, etc.Riferimenti:

http://www.python.org/download/
http://www.python.org/download/download_other.html

Per avere un "assaggio" della sintassi di Python vedi:

http://net.supereva.it/aleax/Python/ItaPythInst.htm

Un esempio della compattezza del linguaggio Python è dato dal seguente programma (il sorgente è completo!). Non sarà il calendario di Aida Yespica, ma meglio di nulla...

```
import calendar
calendar.prcal(2005)
```

Standard library reference:

http://docs.python.org/lib/lib.html

Consiglio a tutti i frequentatori del Forum di familiarizzare con la sintassi del Python: è un linguaggio facile da apprendere e molto versatile ed avrà un'adeguata copertura su ioProgrammo!

SERVIZIO CLIENTI
e-mail: sevizioclienti@edmaster.it
Tel. 02 83 12 12

SOSTITUZIONE CD
Inviare il CD Rom difettoso in busta chiusa a: Edizioni Master Servizio Clienti Via Ariberto, 24 - 20123 (MI)

Creare un sistema completo antispam con C#

Spam Killer

Vedremo come realizzare un sistema completo di filtro AntiSpam con C#, sfruttando SQL Server, C# e WSE 2, iniziando dalla parte Server

Utilizza questo spazio per le tue annotazioni

Ritrovarsi sovente la casella piena di posta indesiderata è ormai una consuetudine. Lo SPAM è indubbiamente un lato negativo di Internet che, giorno dopo giorno, assume delle proporzioni sempre maggiori sia in termini di traffico sviluppato sia di danno economico. In questo articolo realizzeremo un piccolo sistema AntiSpam che ci potrebbe aiutare ad arginare il problema.

IL NOSTRO SISTEMA ANTISPAM

Il sistema che realizzeremo sarà costituito da un server contenente le definizioni, e da un piccolo client il cui scopo sarà duplice: scaricare le e-mail e accedere alle definizioni di segnalazione di e-mail sospette. Il client stesso filtrerà le mail in arrivo, sulla base delle definizioni scaricate dal server in formato XML e marcherà le mail sospette con la dicitura "SPAM". Gli utenti abilitati al servizio segnaleranno al server, direttamente dal client, eventuali mail indesiderate. Il server immagazzinerà l'informazione all'interno di un Database che conterrà le segnalazioni di tutti gli utenti registrati. Ciò che verrà archiviato sul server è solo l'indirizzo del mittente della mail sospetta. Qualora lo stesso indirizzo fosse segnalato da più utenti, verrà incrementato un contatore che definirà il "rating" dell'indirizzo. Più il rating è alto, più è probabile che la mail ricevuta sia effettivamente spam. La lista degli indirizzi "sospetti" verrà scaricata in formato XML dal client di posta che, confrontando gli indirizzi dei mittenti delle mail in arrivo con quelli contenuti nel file delle definizioni, potrà marcare opportunamente le mail riconosciute come spam aggiungendo successivamente il valore del rating. L'interazione tra il client e il server avviene tramite un Web Service predisposto ad accettare le richieste degli utenti abilitati. L'amministrazione del server avviene attraverso un tool di gestione separato da installare sul PC dell'amministratore (**Figura 1**) che fa uso di un secondo servizio web per comunicare con il server.

Fig. 1: *Il tool di amministrazione del server antispam*

Fig. 2: *Schema del Database Anti Spam*

I SERVIZI WEB

Dopo aver realizzato il Database, il primo problema da affrontare è quello di rendere disponibili i dati in esso contenuti verso l'esterno. Il nostro server dovrà permettere di:

- Segnalare una mail come spam
- Trasmettere l'elenco degli indirizzi mail in formato xml
- Registrare nuovi utenti abilitati al servizio
- Eliminare utenti dal Database

Le operazioni appena descritte possono essere divise in due famiglie: gestione utenti e gestione spam. Manterremo quindi la stessa distinzione logica anche sul server, realizzando due distinti Web Service. Per la loro realizzazione possiamo utilizzare i classici servizi Web oppure affidarci a WSE 2.

Avendo la necessità di proteggere l'accesso al servizio, la scelta cade obbligatoriamente su WSE 2 in quanto implementa le specifiche WS-Security. Dopo aver installato WSE 2, avviamo Visual Studio 2003 e creiamo una nuova libreria di classi chiamandola *"AntiSpamServerEngine"* (**Figura 3**) ed al suo interno creiamo subito due nuove classi chiamate "*SpamManager.cs*" ed "*UserManager.cs*" che costituiranno servizi web (**Figura 4**). Nel corso di tutto lo sviluppo utilizzeremo una sola "Soluzione" in cui inseriremo tutti i progetti che compongono il sistema AntiSpam. Questa organizzazione Soluzione /Progetti è principalmente una comodità fornita da molti editor che non influisce sulla realizzazione del progetto stesso. Non disponendo di Visual Studio potremmo semplicemente realizzare tanti progetti separati organizzati come meglio ci pare. Predisponiamo quindi il nostro progetto all'utilizzo di WSE 2 aggiungendo i riferimenti a Microsoft.Web.service2, System.Web e System.Web.Services.

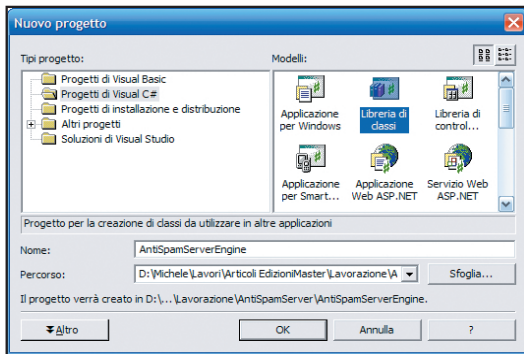



***Fig. 3:* Nuova soluzione in Visual Studio**

SCRIVIAMO IL CODICE DI SPAMMANAGER

Ora che i riferimenti corretti ci sono, iniziamo a costruire le classi. Analizziamo la *SpamManager* e, più precisamente il metodo che ci consente di segnalare un indirizzo di posta da considerare come SPAM.

```
using System;
using System.Data;
using System.Xml;
using ioProgrammo.DataLayer;
using Microsoft.Web.Services2;
using Microsoft.Web.Services2.Messaging;
using Microsoft.Web.Services2.Security;
using Microsoft.Web.Services2.Security.Tokens;
```

Nella prima parte della classe utilizziamo le direttive Using per richiamare in forma abbreviata i metodi delle rispettive classi. Nell'elenco, oltre a quelle standard, sono evidenti quelle relative all'utilizzo di WSE 2.

```
namespace ioProgrammo.AntiSpamServerEngine {
```

Definiamo il namespace e subito dopo il servizio che si chiamerà SpamManager.

```
[SoapService("http://ioProgrammo.Articoli/AntiSpam/")]
public class SpamManager : SoapService {
                    //System.Web.Services.WebService
```

Il metodo si chiama *SegnalaMail* ed accetta come parametro di ingresso la mail da segnalare. L'attributo *[SoapMethod("urn:SegnalaMail")]* non fa altro che comunicare a *SoapService*, che il metodo *SegnalaMail* sarà disponibile via Soap Message. SoapMethod sostituisce in pratica il WebMethod dei servizi web tradizionali. SpamManagerDB è la classe che astrae la logica di accesso ai dati dal nostro servizio web. La vedremo nel prossimo paragrafo.

```
[SoapMethod("urn:SegnalaMail")]
public int SegnalaMail(string EMail){
  SpamManagerDB manager = new spamManagerDB();
  if ( IsValidEmail(EMail) ){
     return manager.AddNewSpamMail(EMail);
  }else{
     throw new ApplicationException("Indirizzo e-mail
                                    non valido");} }
```

Prima di inserire l'indirizzo e-mail nel Database, ci assicuriamo che sia almeno sintatticamente valido, lo facciamo attraverso il metodo *IsValidAmail* che, usando una Regular Expression, ci restituisce *true* se l'indirizzo è sintatticamente corretto, *false* in caso di indirizzo errato.

```
private bool IsValidEmail(string Email) {
return System.Text.RegularExpressions.Regex.IsMatch(
    Email, @"^([\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}
     \.[0-9]{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}|[0-
                                  9]{1,3})(\]?)$"); }
```

È evidente che a questo metodo "manca" qualcosa: la comunicazione con il Database. Prima di rendere disponibili al pubblico i servizi analizziamo la logica di accesso ai dati per capire cosa fa la SpamManagerDB. Nel precedente paragrafo abbiamo visto che, se l'indirizzo mail risulta sintatticamente corretto, viene chiamato:

```
SpamManagerDB manager = new SpamManagerDB();
return manager.AddNewSpamMail(EMail);
```

quindi, un metodo del Data Layer, *AddNewSpamMail*. Analizziamo la struttura di questo metodo:

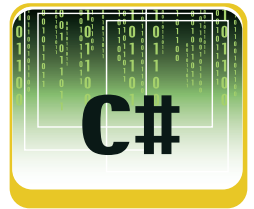
```
public int AddNewSpamMail(string Email){
int Rating = EmailCounter(Email);
  if ( Rating == 0 ){
```

WSE 2 può essere scaricato dal sito http://msdn.microsoft.com/webservices/. L'installazione è abbastanza rapida e consente sin da subito di sfruttare le nuove funzionalità di WSE 2. Il setUp installerà anche una serie di esempi utili a comprendere meglio WSE 2 e le sue potenzialità.

***Fig. 4:* Nuova soluzione in Visual Studio**

```
    return AddNewEmail(Email);
}else if ( Rating > 0 ) {
    return UpdateEmailRating(Email);
}else{
throw new ApplicationException("Errore durante la
                          segnalazione di un URL");} }
```

La prima cosa che andremo a verificare è se la mail è stata segnalata ; facciamolo richiamando il metodo *EmailCounter(…)* passando come argomento la mail che vogliamo controllare, assegnando il valore di ritorno ad una variabile chiamata Rating: se il rating è 0 allora l'indirizzo non è presente nel Data Base. È il caso in cui un indirizzo sia stato segnalato per la prima volta. Il metodo *AddNewEmail(Email))* si occupa di inserire la mail all'interno del DB. Se invece il rating è maggiore di 0, l'indirizzo passato come argomento al metodo *EmailCounter* è già presente nel Database e quindi va incrementato solo il rating. Una volta incapsulati tutti i metodi per l'accesso ai dati nel DataLayer, dobbiamo referenziarlo nel progetto *AntiSpamServerEngine* per poterlo utilizzare. La procedura è identica a quella descritta nel **tutorial**, fatta eccezione per la scheda da scegliere al passaggio 2. Al posto di selezionare gli elementi dalla scheda ".net", li dobbiamo selezionare dalla scheda "Progetti". Negli ultimi due paragrafi abbiamo realizzato la struttura di Web Service, la logica di accesso ai dati ed abbiamo creato gli opportuni riferimenti tra i progetti. È arrivato il momento di rendere accessibili le nostre informazioni.

L'HOST DEL SERVER ANTISPAM

Usando WSE 2 è possibile esporre le classi in svariati modi. Possiamo utilizzare un Server Web insieme al protocollo http, oppure possiamo realizzare un servizio di Windows o un'applicazione (magari di tipo console) da interrogare via TCP. La soluzione più indicata per il nostro server AntiSpam è quella di utilizzare un sito Web. Non dimentichiamoci infatti che potremmo decidere di offrire questo servizio agli utenti che ne facciano richiesta. Esporlo su un sito ci dà il vantaggio di avere un servizio facilmente localizzabile attraverso un url classico (http://....) e, lo stesso sito potrebbe essere usato per promuovere il servizio e far registrare gli utenti. Aggiungiamone quindi uno al nostro progetto chiamandolo *ASService* e referenziamo subito il progetto *AntiSpamServerEngine*. Dobbiamo far sì che questo sito intercetti le richieste SOAP in arrivo e le rimandi alle classi del progetto *UserMamanager* e *SpamManager*. Per farlo, dobbiamo compiere manualmente un paio di operazioni **(vedi tutorial sull'attivazione del servizio Web)**. Come abbiamo visto nel terzo passo, le classi del progetto AntiSpamServerEngine sono ora accessibili attraverso il sito in *ASService*. Tutti i metodi pubblici marcati con l'attributo *[SoapMethod()]* sono descritti nel contratto WSDL e possono essere utilizzati nei progetti. Il primo che ne farà uso, come accennato all'inizio di questo articolo, è il Tool di Amministrazione del servizio AntiSpam. Il tool è abbastanza semplice nella sua implementazione, quindi non verrà trattato approfonditamente se non per un aspetto molto importante: la sicurezza. Sfrutteremo infatti una delle funzionalità di questo programma per descrivere ed implementare un meccanismo di sicurezza personalizzato usando WS-Security.

LA SICUREZZA CON WS-SECURITY

Molto probabilmente, la nostra applicazione risiede su un server web pubblico. Questa caratteristica lo espone a tutti gli utenti che ne conoscono (o ne scoprono) l'indirizzo. Provate a immaginare cosa succederebbe se lasciassimo le sue funzionalità aperte a tutti. Nel paragrafo relativo alla realizzazione del servizio web abbiamo analizzato un metodo *(SegnalaMail)* sprovvisto di sistemi di protezione. Analizzeremo ora una funzionalità del tool di amministrazione che richiede un meccanismo di protezione e vedremo come implementare tale meccanismo sia sul client sia sul server. Iniziamo a creare un nuovo progetto, sempre all'interno della stessa soluzione, questa volta di tipo Windows Forms che chiameremo AdminTool e, seguendo solo il primo passo del secondo tutorial, abilitiamolo all'utilizzo di WSE 2. Aggiungiamo inoltre un riferimento a System.Web e *System.Web.Services* seguendo la stessa procedura descritta nel primo tutorial. Per poter utilizzare i nostri servizi remoti, dobbiamo scrivere del codice che ci consenta di effettuare delle chiamate ai metodi esposti. Tale codice rappresenta il SoapClient che scaturisce dal contratto (WSDL) visto in precedenza. Fortunatamente, esistono dei tool in grado di generare il codice al posto nostro. Per WSE 2.0 il tool si chiama wsewsdl2.exe e si trova nella directory in cui abbiamo installato WSE 2 (generalmente *C:\Programmi\MicrosoftWSE\v2.0 \Tools\Wsdl*). Questo tool accetta diversi parametri in ingresso, quelli che ci interessano in questo momento sono:

- L'url in cui si trova il contratto soap, nel nostro caso *http://localhost/ASServer/UserManager.ashx*
- Un file in cui salvare in codice generato.

La classe creata dal tool è a tutti gli effetti il nostro SoapClient che permette al nostro tool di comunicare con il servizio web. Inseriamo quindi il file salvato in C:\UserManagerServices.cs nel nostro client con una semplice operazione di drag and drop. Il meto-

Le specifiche complete del WS-Security sono raggiungibili dal sito http://msdn.microsoft.com/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnglobspec/html/wssecurspecindex.asp insieme a tutte le altre specifiche della famiglia *WS-**

Maggiori dettagli sulla clausola FOR XML di Sql Server sono spiegati in questo articolo: http://www.aspitalia.com/articoli/db/xmlt-sql.aspx su Aspitalia.com.

L'amministrazione del server AntiSpam avviene tramite il tool AdminTool descritto brevemente nell'articolo. Per usarlo è sufficiente avviarlo ed effettuare il log on. Una volta validate le credenziali, si potrà selezionare una delle schede disponibili ed effettuare le relative operazioni.

do che utilizzeremo per l'esempio è quello che si occupa dell'inserimento di un nuovo utente all'interno del database remoto. Iniziamo però a vedere come è fatta la parte server di questo metodo (situata nel progetto AntiSpamServerEngine nel file UserManager.cs):

```
[SoapMethod("urn:AddNewUser")]
public int AddNewUser(UserInfo userInfo){
```

Come al solito, abbiamo la dichiarazione del [*SoapMethod()*] e quella del metodo stesso (*AddNewUser()*) che accetta come parametro in ingresso un tipo complesso, ovvero un semplice oggetto che descrive l'utente che vogliamo inserire nel Data Base. Leggiamo innanzitutto il SoapContext del messaggio soap in arrivo. Descrivere precisamente cosa sia il context esula dallo scopo di questo articolo. Volendo sintetizzare, possiamo vederlo come un meccanismo per applicare dei filtri al messaggio soap in uscita.

```
SoapContext context = RequestSoapContext.Current;
if ( context == null ){
   throw new ApplicationException("Sono ammesse
                                  solo richieste SOAP");}
```

Il primo controllo che facciamo è relativo alla presenza o meno di un context. Come vedremo più avanti infatti, le credenziali dell'utente per effettuare il login vengono passate appunto nel context. Il controllo appena effettuato taglia tutte le richieste che non lo hanno.

```
foreach (SecurityToken token in context.Security.Tokens){
    UsernameToken userT = (UsernameToken)token;
    if ( userT.Principal.IsInRole("Admin") ){
```

Il secondo tipo di controllo effettuato ha lo scopo di ciclare all'interno di tutti gli eventuali context presenti nel messaggio, recuperare il token con le credenziali dell'utente e verificarne il ruolo. In questo esempio specifico, vogliamo consentire l'utilizzo di questo metodo ai soli membri del gruppo Admin.

```
   user = new UserManager();
   return user.AddNewUser(userInfo.UserName,
    userInfo.Password, userInfo.Email, userInfo.Livello);
}else{
   throw new ApplicationException ("Utente non
                                    autorizzato"); } }
   return 0; //non dovrebbe mai arrivare qui }
```

Il resto del codice si occupa semplicemente di eseguire il metodo del DataLayer per l'inserimento dell'utente o, in caso di assenza di permessi sufficienti, di far tornare una eccezione. Ma dove avviene la verifica reale dei permessi? E dove viene associato il ruolo?

GESTIAMO L'AUTENTICAZIONE

Il comportamento di default del sistema di autenticazione si basa sugli account del sistema operativo. Questo vuol dire che è necessario creare tanti utenti sul server quanti sono quelli abilitati all'utilizzo del servizio. Inutile dire che, se il servizio deve essere esteso a molti utenti, questa scelta è impensabile. Fortunatamente però, abbiamo la possibilità di creare un nostro meccanismo di autenticazione personalizzato. Creiamo dunque un nuovo proget-

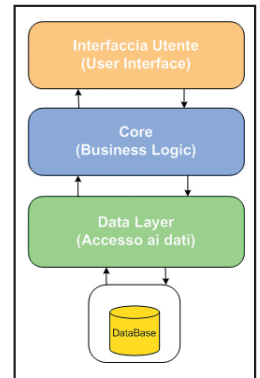
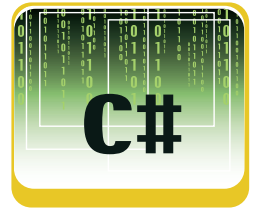
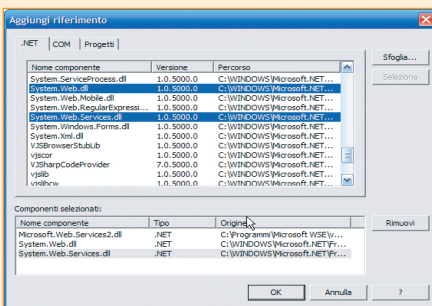
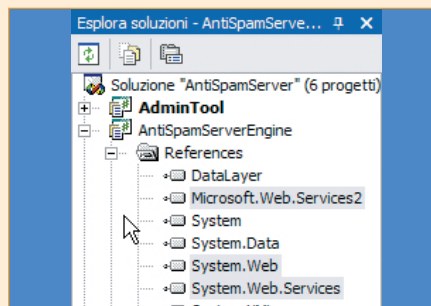
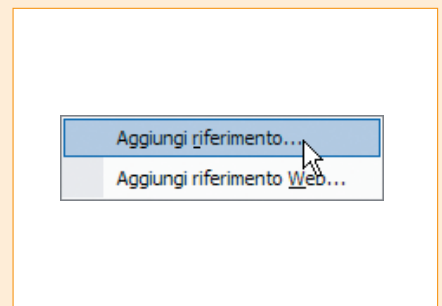
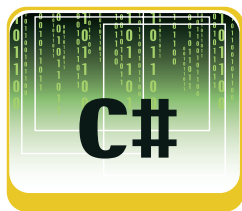



Fig. 5: *Schema di applicazione strutturata a livelli*

Fig. 6: *Il tool wsewsdl2.exe per la creazione del SoapClient*

AGGIUNTA DEI RIFERIMENTI AL PROGETTO

1 Selezioniamo la cartella *References* nell'area *Esplora Soluzioni* del nostro progetto (vedi Fig. 4). Clicchiamo con il tasto destro del mouse e selezioniamo la voce "Aggiungi Riferimento". Questa operazione ci consentirà di referenziare altri assembly nel nostro progetto.

2 Dalla maschera *Aggiungi riferimento* selezioniamo la scheda *.net* (dobbiamo referenziare gli assembly del *.net framework*) e scegliamo dall'elenco *Microsoft.Web.service2, System.Web* e *System.Web.Services.* Una volta selezionati, potremmo utilizzare tutte le funzionalità.

3 Dopo aver confermato la selezione eseguita al passo precedente, gli assembly referenziati saranno visibili nella cartella *References* del nostro progetto. Utilizzando la direttiva *Using* nel nostro codice, possiamo accedere direttamente alle funzionalità esposte.

Il progetto *AntiSpamServerEngine* è una libreria di classi, quindi priva di interfaccia utente, ma abbiamo comunque deciso di esporre il nostro servizio attraverso un sito web. Farlo è possibile grazie a WSE 2. *SoapService* (usato per realizzare i servizi Web) deriva infatti da *SoapReceiver* che implementa l'interfaccia *IHttpHandler* (la stessa che implementano le normali pagine Asp.net). L'indirizzo a cui le nostre classi saranno raggiungibili lo definiamo attraverso un *HttpHandler* direttamente nel *web.config* del progetto Web.

to (CustomSecurityToken Manager) che avrà lo scopo di sostituire il meccanismo di autenticazione di default di WSE 2 con uno nostro. Avendo predisposto una tabella nel Database per immagazzinare i dati degli utenti, valideremo le credenziali ricevute direttamente sul nostro database. Un progetto separato ci offre il vantaggio non indifferente di poter sostituire il meccanismo di autenticazione in qualsiasi momento, senza dover intervenire sul resto della soluzione. *CustomSecurityTokenManager* estrarrà i dati dell'utente che "chiama" il servizio dal contex accennato nel precedente paragrafo. I dati estratti (nome utente e password) verranno confrontati con quelli contenuti nel Database e, in caso di riscontro positivo, la chiamata al servizio potrà essere effettuata. L'unica classe che CustomSecurityTokenManager contiene è strutturata come segue:

```
protected override string AuthenticateToken(
                                UsernameToken token) {
```

in cui viene sostituito il comportamento di default con quello personalizzato

```
  string Livello = CheckUser(token);
```

questo metodo non fa altro che far tornare il livello di abilitazione dell'utente che ha fatto il log on dal Data Base (vedi commenti nel codice) e …

```
if ( Livello != string.Empty ){
    string[] roles = {Livello};
    token.Principal = new GenericPrincipal(new
             GenericIdentity(token.Username), roles);
    return token.Password;
```

assegnarlo all'utente che stà chiamando il metodo.

```
  }else{
    //se il livello torna vuoto, il login non è stato possibile
    return string.Empty; }
}
```

Per rendere attivo il nostro meccanismo di autenticazione personalizzato, dobbiamo aggiungere un riferimento a questa classe nella sezione security del nodo <*Microsoft.Web.Service2*> del *web.config*

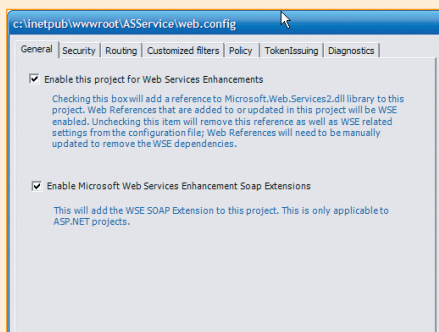
```
<security>
   <securityTokenManager type=
          "ioProgrammo.CustomSecurityTokenManager
      .CustomValidator, CustomSecurityTokenManager"
   xmlns:wsse="http://docs.oasis-open.org/wss/2004/
      01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
                   qname="wsse:UsernameToken" />
</security>
```

Il nostro sistema di autenticazione custom è ora pronto: salviamo il progetto e chiudiamolo. Per completare l'intera procedura dobbiamo ora effettuare delle interrogazioni protette ai nostri web service.

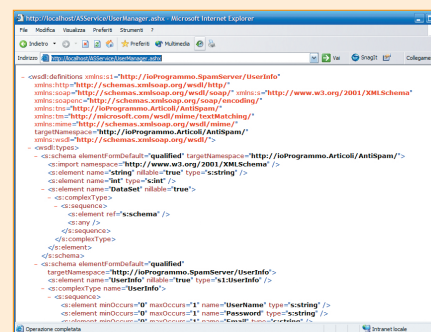
L'INTERROGAZIONE PROTETTA: PASSIAMO LE CREDENZIALI

Nei 2 precedenti paragrafi abbiamo:

1. protetto i metodi del servizio web verificando il "ruolo" dell'utente che ha chiamato il metodo.

ATTIVAZIONE DEL SERVIZIO WEB IN WSE 2

1 **Facendo click con tasto destro del mouse sul progetto web (ASService), apparirà un menu la cui ultima voce è "WSE Settings 2.0" che aprirà la maschera in figura. Attraverso questo tool possiamo abilitare correttamente il nostro progetto all'utilizzo di WSE. Spuntiamo i due elementi come in figura e confermiamo.**

```
<httpHandlers>
<add type="ioProgrammo
        .AntiSpamServerEngine.SpamManager,
             AntiSpamServerEngine" path=
          "SpamManager.ashx" verb="*" />
<add type="ioProgrammo
   .AntiSpamServerEngine.UserManagerServic
           e, AntiSpamServerEngine" path=
            "UserManager.ashx" verb="*" />
</httpHandlers>
```

2 **Nel web.config dobbiamo configurare due http handler per reindirizzare le richieste http alle nostre classi. Usiamo quindi due nomi fittizi *(UserManager* e *SpamManager)* associandoli all'estensione .ashx già mappata su IIS. Questa configurazione permette di girare le richieste alla pagina UserManager.ashx alla nostra classe *ioProgrammo.AntiSpam ServerEngine.UserManagerService* che rappresenta il servizio web.**

3 **Una volta salvato il web.config e compilato, se proviamo a richiamare la pagina http://localhost/ASService /UserManager.ashx (che abbiamo precedentemente mappato), ci ritornerà la pagina in figura. Quello che vediamo nella pagina è il "contratto soap" (WSDL) esposto dal servizio web (quindi dalla nostra classe). Tale contratto verrà poi utilizzato per creare i metodi per interrogare il servizio.**

2. creato un nostro sistema di autenticazione personalizzato per sostituire quello di default di WSE 2.

Quello che ci manca è inviare le credenziali dell'utente, per permettere al nostro CustomSecurityTokenManager di controllarle. Per analizzare questa funzionalità prendiamo ad esempio un metodo del tool di amministrazione del servizio web, più precisamente la funzionalità di inserimento di un nuovo utente (*btnSaveUser_Click…*).
La prima cosa da fare è predisporre il "messaggio" che deve essere inviato al servizio web. Sebbene la classe proxy creata con il tool *wsewsdl2.exe* dovrebbe consentire di effettuare l'interrogazione al servizio in modo diretto (chiamando solo un metodo di tale classe), nel nostro esempio ci occuperemo noi di predisporre il messaggio soap da inviare. Questo ci consente di comprendere meglio cosa accade in messaggio soap. Iniziamo col definire il nostro messaggio che sarà di tipo SoapEnvelope. Il corpo di questo messaggio conterrà l'oggetto userInfo creato sulla base delle informazioni dell'utente che vogliamo inserire nel Data Base:

```
SoapEnvelope envelope = new SoapEnvelope();
envelope.CreateBody();
envelope.SetBodyObject(userInfo);
```

Non dimentichiamoci che dobbiamo passare le credenziali dell'utente che deve eseguire l'operazione di inserimento del dato. Stiamo chiamando un metodo protetto ricordate? Come detto nel precedente paragrafo, le credenziali vanno passate nel context del messaggio quindi:

```
envelope.Context.Security.Tokens.Add(new
        UsernameToken(_UserName, _Password,
            PasswordOption.SendPlainText));
```

Il token di autenticazione viene costruito con i dati inseriti nel form di log in del tool di gestione. Il token verrà poi inserito nel context del messaggio e recuperato dal server con la procedura che abbiamo descritto in precedenza.
Nelle restanti righe di codice non facciamo altro che inviare il nostro messaggio soap al servizio web ed interpretarne la risposta.

```
envelope.Context.Addressing.Action = new Action(
                        "urn:AddNewUser");
UserManager userManager = new UserManager();
int RetVal = userManager.AddNewUser(envelope);
```

Tutti gli altri metodi protetti utilizzano lo stesso meccanismo.

IL DOWNLOAD DELLE DEFINIZIONI

L'ultima cosa che manca per completare il server AntiSpam è il download delle definizioni in formato XML. È superfluo descrivere passo passo il SoapMethod che si occupa di recuperare i dati dal Database in quanto molti degli aspetti (DataLayer, WS-Security ecc.) sono stati già trattati nel resto dell'articolo. È importante però segnalare un paio di passaggi interessanti. Sappiamo che il client utilizzerà un file XML come file di definizione delle mail di spam. Tale file deve essere generato sul server e salvato direttamente sul PC dell'utente.
Per la generazione del suddetto file possiamo utilizzare una comoda funzione di Sql Server: FOR XML. Si tratta di una variante del linguaggio SQL Standard permette a Sql Server di restituire un result set direttamente in formato XML.
Nel DataLayer abbiamo un metodo chiamato *MailList()* che interroga il Data Base con la seguente stringa:

```
string Sql = "Select [Email], [Rating] from tblEmail
    order by [Rating] desc for xml auto, elements";
```

Il result set generato è composto da tag XML e verrà inviato al client in formato stringa. Una volta ricevuto dal client, la stringa deve essere estratta dal messaggio soap ed utilizzata per creare l'XML con le definizioni.

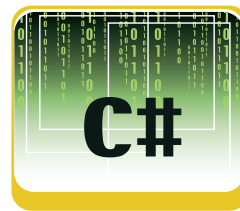
```
SpamManager sManager = new SpamManager();
XmlDocument doc = new XmlDocument();
doc.LoadXml("<?xml version=\"1.0\" encoding=
        \"utf-8\" ?><SpamDefinition>" + sManager
    .GetSpamList(envelope) + "</SpamDefinition>");
```

per poi essere salvato sull'Hard Disk.

```
doc.Save(Environment.CurrentDirectory +
                    @"\Definition.xml");
Console.WriteLine("File delle definizioni salvato in " +
  Environment.CurrentDirectory + @"\Definition.xml");
```

CONCLUSIONI

In questo primo articolo abbiamo analizzato la problematica legata ai filtri AntiSpam e ne abbiamo realizzato uno abbastanza semplice. Le tecnologie utilizzate per la realizzazione del sistema sono tra le più recenti disponibili ad oggi (vedi WSE 2). Nel prossimo numero vedremo realizzare un client di posta che, sfruttando sempre WSE 2 per "parlare" con il server, possa recuperare le definizioni ed usarle per filtrare la mail. Alla prossima puntata!

Michele Locuratolo

INSTALLAZIONE E TESTING

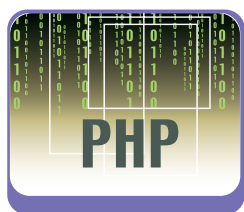
Il primo passo per da effettuare per utilizzare l'applicazione è creare il Data Base. Usando il vostro tool di gestione preferito per Sql Server o MSDE, create un nuovo Data Base ed eseguite le istruzioni Sql contenute nel file *DB_AntiSpam.sql*. Copiate successivamente la cartella ASService sul vostro Hard Disk e rendetela una directory virtuale in IIS. Per testare il servizio web useremo il tool di Amministrazione. Una volta avviato, digitiamo *User Name* e password nella form di LogIn e clicchiamo sul relativo pulsante. Una volta autenticati, possiamo eseguire una qualsiasi delle operazioni per cui il tool è preposto: inserimento di nuovi utenti, visualizzazione e cancellazione di utenti, visualizzazione delle segnalazioni di spam. Tutte queste funzionalità sono accessibili dalle relative schede.

L'AUTORE

L'autore può essere contattato attraverso il suo blog su http://blogs.mindbox.it e sarà lieto di rispondere alle domande dei lettori.

La programmazione orientata agli oggetti in PHP5

PHP5, la terza generazione

Se siete programmatori PHP dovreste leggere questo articolo; se non lo siete dovreste leggerlo ugualmente: comprenderete infatti la logica della programmazione ad oggetti

Avete mai provato a inserire la parola PHP nel motore di ricerca Google? Il risultato è sorprendente! 760.000.000 di pagine trovate. Più del doppio di ASP, sette volte tanto JSP e circa 10 volte tanto ASPX. Ora, chiaramente, questo è un metodo piuttosto empirico, la diffusione di un linguaggio non è data da queste informazioni, ma è anche vero che questo esperimento dà la misura di quanto PHP abbia inciso sullo sviluppo di applicazioni per il web negli ultimi anni.
Se si pensa che PHP5 è stato ufficialmente rilasciato da pochi mesi, è altrettanto evidente che la parte del leone in questa enorme diffusione l'hanno fatta PHP3 e PHP4. Come mai si è scelto di rivoluzionare completamente PHP5 rendendo in parte le vecchie applicazioni non compatibili con la nuova versione? La risposta è abbastanza semplice e si identifica nella necessità di dotare PHP5 di un modello ad oggetti degno di un linguaggio moderno. La programmazione a oggetti consente infatti una straordinaria flessibilità nel modificare globalmente progetti di dimensioni enormi cambiando poche righe.

IL VECCHIO MODELLO PROCEDURALE

Il funzionamento di un modello procedurale è noto a tutti ed a maggior ragione noto ai programmatori PHP. Molto semplicemente si definisce una funzione che esegue una qualche azione

```
<?
function sportello($operazione) {
        if ($operazione = 'apri') {
                $statosportello='aperto';
        } else {
                $statosportello='chiuso'; }
        return $statosportello; };
$macchina = sportello('apri');
?>
```

Si tratta di una funzione semplice. Riceve una stringa contenente un comando e ritorna uno stato. Funziona perfettamente, ma ha dei limiti. Prima di tutto non viene salvato da nessuna parte lo stato dello sportello della macchina, aperto o chiuso. Sarebbe possibile salvare questo stato in una variabile globale per interrogarla successivamente, avendo però più di una macchina da gestire avremmo bisogno di piu di una variabile. Un'ulteriore soluzione sarebbe creare una matrice macchine X stati dove a ciascuna macchina corrisponde uno stato dello sportello, per poi gestirla con un ciclo di *for*. Esiste qualche altra limitazione, ovvero in realtà non c'è un legame forte fra la variabile macchina e la *function sportello*. Il programmatore che ha sviluppato il codice sa che quella funzione serve a gestire delle variabili che sono delle macchine. Ma nulla vieta di creare una variabile "barca" e farla gestire alla funzione "*sportello*", se un secondo programmatore dovesse manutenere il codice dovrebbe intuire la logica di chi ha sviluppato per primo il programma e se casualmente il nome delle variabili non fosse significativo potrebbe accadere veramente di tutto. Infine volendo adottare un comportamento specifico per una macchina particolare dovremmo riscrivere interamente la funzione.

IL NUOVO MODELLO AD OGGETTI

Proviamo a fare un ragionamento molto semplice. Prima di tutto il nostro programma opera su un insieme ben preciso, ovvero le macchine. Possiamo dire che ogni macchina possiede almeno uno sportello e che questo sportello può esse-

re aperto o chiuso. Ora, provando a trasferire questo ragionamento in termini di programmazione PHP ad oggetti avremmo qualcosa del genere.

```
<?
class macchina {
  var $stato_sportello;
  function macchina() {
    global $stato_sportello;
    $this->stato_sportello ='chiuso'; }
  function sportello($operazione) {
    if ($operazione='apri') {
      $this->stato_sportello='Aperto';
    } else  {
      $this->stato_sportello='Chiuso';
    }
  }
};
$fiat_500 = new macchina;
$fiat_500->sportello('apri');
echo 'Lo sportello della tua Fiat 500 è
                              '.$fiat_500->stato_sportello;
?>
```

In un primo momento abbiamo definito la categoria delle macchine. Da questo momento in poi quando parleremo di una categoria ben definita da un'insieme di caratteristiche omogenee ci riferiremo ad essa con il termine "Classe". Perciò abbiamo nella clausola *class macchina {...}* definito la classe delle macchine, dotata della caratteristica comune di possedere uno sportello il cui stato può essere aperto o chiuso. Avendo definito la classe macchina possiamo ben dire che la Fiat 500 fa parte dell'insieme delle macchine. Da questo momento in poi quando vorremo indicare che un elemento appartiene ad una classe, ci riferiremo ad esso con il termine "Oggetto". Perciò la Fiat 500 è un Oggetto della Classe macchina. Nel nostro esempio la riga

```
$fiat_500 = new macchina;
```

Indica che la Fiat 500 è un oggetto della classe macchina. Come tale aderisce ai comportamenti definiti in questa classe. In particolare quando un nuovo oggetto viene inizializzato, viene messa a sua disposizione un'intera struttura che racchiude tutti comportamenti della classe. La riga

```
$fiat_500->sportello('apri');
```

utilizza un comportamento tipico della classe macchina per eseguire un'azione, cioè quella di variare lo stato dello sportello da aperto a chiuso.
Infine

```
echo 'Lo sportello della tua Fiat 500 è '.$fiat_500->
                                         stato_sportello;
```

Lo stato dello sportello è sempre un attributo della classe macchina, in questo caso associato in particolare alla nostra Fiat 500. Da questo momento in poi chiameremo "Metodi" quelli che fino ad ora abbiamo chiamato semplicemente "comportamenti di una classe", e "Campi" quelli che fino ad ora abbiamo inteso come attributi di una classe. Perciò *sportello()* è un metodo della classe macchina e *stato_sportello* è un campo della classe macchina. Infine l'oggetto Fiat 500 richiama un metodo tipico della classe macchina.

MATTONE DOPO MATTONE

Come in tutte le cose, un oggetto non esiste fino a quando qualcuno non lo costruisce. Perciò supposto che esista il progetto di una macchina, certamente questa macchina non esiste finché qualcuno con le proprie mani non comincia a montarne i pezzi. Perciò la nostra Classe macchina è un po' come il progetto che un ingegnere fa su carta, ma non esiste nessuna macchina di quel tipo fino a che qualcuno non la costruisce. L'oggetto inizia ad avere vita con l'istruzione new. Nel nostro esempio:

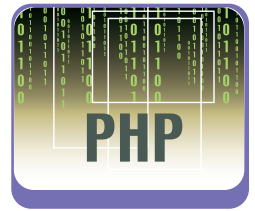
```
$fiat_500 = new macchina;
```

Questa istruzione crea l'oggetto. Inizializza in memoria una struttura dati adatta a contenerlo. In particolare viene richiamato il "Costruttore" della classe. In PHP4 il costruttore corrisponde alla funzione che ha il nome identico a quello della classe. Nel nostro esempio il costruttore setta a "Chiuso" lo stato dello sportello. Infatti, qualunque macchina appena uscita di fabbrica ha tipicamente gli sportelli chiusi.

DIFFERENZE CON IL METODO PROCEDURALE

Mentre nel metodo procedurale avremmo dovuto creare una matrice per contenere lo stato di ciascuna macchina, in questo caso ci basta creare un oggetto di tipo macchina.
I campi ed i metodi dell'oggetto appena creato, incluso il suo stato, verranno salvati in memoria. In sostanza

```
$fiat_500 = new macchina;
$fiat_panda = new macchina;
```

Indicano ciascuno un oggetto di tipo diverso che ha uno stato sportello diverso.

CLONARE GLI OGGETTI

Considerate un oggetto

```
$Fiat_500 = new macchina
```

e considerate l'ipotesi di volere duplicare il contenuto dell'oggetto, come spesso si fa per salvare uno stato in memoria, modificare dei valori e poi utilizzare i valori salvati per ottenere dei risultati. L'idea più semplice sarebbe fare qualcosa del genere.

```
$Fiat_500 = Fiat_500_Temp
```

Sappiate che questa istruzione non funziona! Di fatto crea un nuovo riferimento allo stesso oggetto ma non lo duplica. Perciò il modo corretto di duplicare un oggetto è utilizzare la funzione clone

```
$oggetto2=clone(
              $oggetto1)
```

Non è necessario ricreare una struttura apposita per contenere gli stati. A ciascun oggetto corrisponde una struttura separata, definita dalla classe macchia, i metodi associati ad un oggetto influiscono solo su quell'oggetto. I campi dell'oggetto sono influenzati solo dal metodo associato a quell'oggetto particolare, e non ad altri.
Un bel vantaggio rispetto alla programmazione procedurale, non trovate? Se la classe fosse definita in un file di include, il programmatore non avrebbe neanche bisogno di sapere come i suoi metodi sono definiti, avrebbe solo bisogno di sapere come questi metodi devono essere usati. Questo è esattamente il miglior modo di manutenere un progetto di grandi dimensioni.
Un programmatore definisce le classi, i metodi, i campi che di ciascuna classe. Altri programmatori usano classi, metodi e campi per programmare le proprie applicazioni secondo certe specifiche, ma non hanno necessità di sapere come i metodi della classe sono programmati. Se i programmatori delle classi trovano un algoritmo migliore e ottimizzato per compiere un'azione, i programmatori che usano le classi non devono cambiare il proprio codice! Loro usano solo i metodi, quello che c'è dietro non è importante per loro.
Segreto svelato!

AUTOCARICARE LE CLASSI

Chiaramente per potere utilizzare una classe all'interno di un progetto è necessario includerla nel proprio progetto. Come già detto la logica della programmazione a oggetti è sfruttata al 100% se classi e programmi che le usano sono definite in file separati. Questo consente eventualmente di modificare l'intero comportamento di un programma semplicemente modificando la classe a cui fanno riferimento gli oggetti che vengono utilizzati all'interno del software. Perciò il modo migliore sarebbe sviluppare le classi in file separati e poi includerle nei propri progetti con un semplice: include *'nome_classe.php'* Questo implica che tutti i file del progetto includano tutti i file delle classi, il che per un numero di classi abbastanza elevato è piuttosto noioso. L'alternativa sarebbe creare un solo file contenente tutte le classi e includere solo quello all'inizio di ogni progetto. Ma questo rende molto meno manutenibile il codice, infatti il caso tipico di sviluppo di progetti in team è tale che ciascun programmatore sviluppi in autonomia le proprie classi su indicazione di un *'funzionale'*.
La soluzione ideale è utilizzare una funzione *__autoload.*
La funzione *__autoload* prevede che i file che definiscono le classi siano identificati con il nome della classe che definiscono. Ad esempio il file che definisce la classe macchina sarà nominato *macchina.php.*
I programmatori possono ridefinire *__autolad* come segue:

```
<?php
function __autoload($class_name) {
   require_once $class_name . '.php';
}
$obj  = new macchina();
$obj2 = new camion();
?>
```

la funzione *__autoload* viene richiamata automaticamente quando un nuovo oggetto viene creato, e come potete vedere dall'esempio include nel progetto il file corrispondente alla classe utilizzata dall'oggetto. Questo automatizza l'inclusione delle classi all'interno dei file.

PHP4 VS PHP5

Ora, vi voglio svelare un piccolo segreto. Quanto qui visto rispetto alla programmazione ad oggetti era applicabilissimo anche in PHP4. Ok, sento fin qui l'eco delle vostre lamentele per aver dichiarato fin qui quant'è bello e quanto è forte PHP5 ed in realtà fa le stesse cose di PHP4. E qui invece vi sbagliate. Il modello ad oggetti di PHP5 è profondamente migliorativo rispetto al precedente. Di fatto vengono introdotte alcuni concetti particolarmente interessanti

- Visibilità/Scopo dei campi e dei metodi
- Dichiarazioni separate per i costruttori
- Introduzione dei distruttori
- Clonazione di oggetti
- Interfacce

Se questa roba vi sembra astrusa, scoprirete abbastanza presto che non lo è affatto. Il più delle volte si tratta di ricondurre problemi reali ad una sintassi programmatica. In realtà il livello di difficoltà logico non è maggiore di quello che incontriamo nel fare dei ragionamenti giornalieri a qualsiasi livello. Vi sentite confortati da questa affermazione? Spero tanto di sì, perché sto tentando di piaggiarvi allo scopo di non farvi demordere. Siete pronti? Diamoci dentro allora!

VISIBILITÀ DI CAMPI E METODI

Togliamoci subito il pensiero e diciamo che la parola "Scope" inglese indica tipicamente l'intervallo di visibilità/utilizzabilità di un elemento. Per ritornare all'esempio della macchina, quando la mattina accendete la macchina girando la chiave, voi date semplicemente un comando e vi aspettate che il motore si accenda. Quello che succede all'interno vi interessa poco. Non comandate direttamente *le candelette*, delegate questo compito a un meccanismo interno che in un qualche modo provoca l'azione che desiderate. Proviamo a vedere come questo si riflette sulla nostra programmazione

```
<?
class macchina {
  private $s_motore = 'Spento';
  public function stato_motore() {
     return $this->s_motore; }
  public function accendi_motore()
  { $this->s_motore='Aperto';}
  public function chiudi_motore() {
    $this->s_motore='Spento'; } };
$fiat_500 = new macchina;
$fiat_500->accendi_motore();
echo 'Il motore della tua 500 è '.$fiat_500->
```

```
                    stato_motore();
?>
```

In questo esempio notate che non andiamo a settare direttamente la variabile *s_motore* che contiene lo stato del motore. Questa variabile è infatti "private", il nostro oggetto non è in grado di utilizzarla direttamente, piuttosto deleghi il compito a un metodo che fisicamente setta lo stato della variabile. Possiamo poi interrogare il contenuto della variabile tramite l'apposito metodo *stato_motore* che ancora una volta ci fa da tramite.

EREDITARIETÀ DELLE CLASSI

Il punto è che la classe macchina è una classe generica. All'interno della classe delle macchine potrebbe esserci una sottoclasse che eredita tutti i comportamenti di una macchina normale, ma ne aggiunge altri che non sono tipici della classe base. Supponiamo ad esempio di volere definire la classe delle macchine Diesel. Se dovessi riscrivere completamente la classe, certamente l'utilità della programmazione a oggetti sarebbe fortemente messa in discussione. Posso invece scrivere una classe che ha tutti i metodi e i campi della sua classe base e in più ne aggiunge o ne modifica quelli principali, ad esempio:

```
class macchina_diesel extends macchina {
  function riscalda_candelette() {
    $this->s_motore='In accensione';} }
```

La sintassi è chiara, la classe macchina_diesel estende le capacità della classe base aggiungendo il metodo *riscalda_candelette* che mette lo stato del motore "in accensione" pur senza accenderlo.
Posso provare questa classe utilizzando

```
$nissan_micra = new macchina_diesel;
$nissan_micra->riscalda_candelette();
echo 'Il motore della tua 500 è '.$fiat_500->
                    stato_motore();
```

Eseguendo questo codice vi accorgerete di un errore. Cioè lo stato del motore rimane sempre "Spento" e non passa mai a riscaldato come dovrebbe. Ma non avevamo raccontato la storia dell'ereditarietà etc? Per cui il campo s_motore non doveva essere ereditato? Esatto non vi sbagliate, il campo s_motore doveva essere ereditato, se non che i campi di tipo "private" non sono ereditabili! Bella fregatura direte voi! E invece no, perché per far sì che un campo che non si vuole rendere pubblico sia comunque ereditato è sufficiente dichiararlo come "protected". Per cui nella classe base la dichiarazione corretta avrebbe dovuto essere

```
class macchina {
        protected $s_motore = 'Spento';
[...]
```

Fatto questo l'avremmo tranquillamente ereditato e avremmo potuto modificarlo a nostro piacimento.
C'è un quarto modo di dichiarare la visibilità di un campo. È possibile dichiarare un campo come "final", in questo caso il campo è ereditabile ma non modificabile dalle classi che lo ereditano.

GESTIRE GLI ERRORI

Fin qui abbiamo controllato se lo sportello era aperto, abbiamo acceso il motore e controllato che fosse acceso. Nulla abbiamo fatto per verificare che il finestrino fosse aperto. Questo non toglie che un programmatore che usa le nostre classi ma che ha fatto abuso di birra, straordinaria compagna dei programmatori, si metta in testa di volere usare un codice del genere:

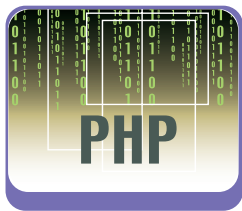
```
$nissan_micra = new macchina_diesel;
$nissan_micra->finestrino='chiuso';
```

In realtà non esiste non nessun campo finestrino, perciò un codice del genere non ha nessun senso. Tuttavia se provate a eseguire il codice non avrete nessun errore. Possiamo gestire questa situazione con l'overload dell'operatore __set(). L'operatore __set() è responsabile fra le altre cose dell'assegnazione del valore di una variabile a un campo. Tutte le classi ereditano da una classe base, fare l'overload dell'operatore __set() significa semplicemente diversificarne il comportamento rispetto a quello che viene fornito di default dalla classe base.
Effettueremo l'overload dell'operatore __set() nella classe macchina, cosi che sia ereditato anche dalla classe macchina Diesel, la sintassi è la seguente:

```
function __set($propName, $propValue) {
  echo "Errore: il campo <b>$propName</b> non esiste";}
```

Se adesso provate ad eseguire l'esempio scoprirete che viene generato l'errore personalizzato secondo le vostre esigenze. Allo stesso modo è possibile effettuare l'overload dell'operatore __get che viene utilizzato per recuperare il valore di una variabile da un campo o tramite un metodo.

CLASSI ASTRATTE

Sulla visibilità dei campi abbiamo già detto. I metodi ricalcano esattamente lo stesso modello di visibilità, eccetto che per le classi astratte. Una classe è

astratta quando dichiara al suo interno dei metodi ma non ne stabilisce le funzionalità, lasciando questo compito invece alle classi che derivano direttamente dalla principale. Ad esempio

```
abstract class macchina
{  abstract function accendi_motore();
   abstract function chiudi_motore();
   abstract function apri_sportello();
}
```

in questo modo dichiariamo solo i metodi all'interno della classe. Sarà compito delle classi derivate eventualmente effettuare l'overload e stabilire le funzionalità dei metodi. Chiaramente in una classe astratta sarà possibile ancora definire dei metodi non astratti. I metodi non astratti saranno ereditati dalle classi derivate. Se in una classe c'è almeno un campo abstract, l'intera classe dovrà essere dichiarata come abstract.

FUNZIONI DI SUPPORTO

Esistono una serie di funzioni che sono correlate alla programmazione a oggetti e possono essere utilizzate per una serie infinita di scopi, ad esempio:

class_exists(string nome_della_classe)	restituisce TRUE se una classe esiste
get_class()	restituisce il nome della classe di appartenenza di un oggetto
get_class_methods(string nome_della_classe)	restituisce i metodi contenuti nella classe in un array
get_class_vars(string nome_della_classe)	restituisce i nomi dei campi definiti in una classe
get_declared_classes()	restituisce un array contenente l'elenco delle classi contenute nello script corrente
get_object_vars(object object)	restituisce un array contenente i campi disponibili per l'oggetto
get_parent_class(mixed object)	restituisce il nome della classe da cui la classe utilizzata dall'oggetto è derivata
is_a()(object object, stringa nome della classe)	restituisce true se un oggetto appartiene a una certa classe passata come parametro alla funzione, oppure ad una classe derivata.
method_exists(object object, string nome_metodo)	Restituisce TRUE se il metodo è disponibile per l'oggetto

COSTRUTTORI E DISTRUTTORI

Sui costruttori abbiamo gia detto qualcosa a inizio articolo. È importante però specificare che in PHP5 i costruttori devono essere esplicitamente dichiarati, inoltre è stato aggiunto il supporto ai distruttori. Inoltre dobbiamo dire qualcosa su come si comportano costruttori e distruttori quando vengono ereditati.
In particolare per dichiarare un costruttore la sintassi è la seguente:

```
function __construct($v_motore) {
   $this->s_motore=$v_motore;}
```

Il costruttore esattamente come nel primo caso che abbiamo visto viene eseguito quando l'oggetto viene inizializzato la prima volta. Per come l'abbiamo dichiarato noi, riceve anche una variabile che viene utilizzata per settare lo stato del motore quando l'oggetto viene inizializzato, ad esempio:

```
$fiat_500 = new macchina('aperto');
echo 'Il motore della tua 500 è '.$fiat_500->
                                   stato_motore();
```

I distruttori vengono invece richiamati viene eliminato ogni riferimento all'oggetto, ed in particolare, ovviamente quando lo script termina;

```
function __destruct() {
   echo "Oggetto distrutto";}
```

POLIMORFISMO

Quando ho sentito questa parola per la prima volta, confesso di avere immaginato il logo di PHP che si contorceva trasformandosi, subendo una metamorfosi come nella migliore delle tradizioni fantascientifiche. L'idea è la seguente. Ci sono classi di elementi che condividono comportamenti e caratteristiche "quasi" identiche, ma non così identiche da renderli completamente omogenei. Si parte da una struttura identica, ma si definiscono poi comportamenti personalizzati. Ad esempio tutte le persone scrivono. Ma in Europa è tipico scrivere da sinistra verso destra, mentre nei paesi arabi è tipico scrivere da destra verso sinistra.

```
interface IPersona {
   function scrivi();}
class arabo implements IPersona {
   function scrivi() {
      echo "sto scrivendo da destra verso sinistra"; } }
class europeo implements IPersona {
   function scrivi() {
      echo "sto scrivendo da sinistra verso destra"; }
}
```

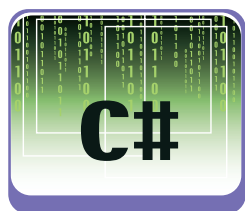
La differenza con le funzioni Abstract, che abbiamo visto prima, è sottile. Ma le classi possono implementare interfacce multiple, mentre non è possibile implementare classi astratte multiple.
Ad esempio è corretta la dicitura:

```
class europeo implements IPersona, IEuro, ICee {}
```

Ammesso che IPersona, IEuro e ICee fossero interfacce.

Semplici accorgimenti per evitare danni irreparabili

Difenderci dagli attacchi di SQL-Injection

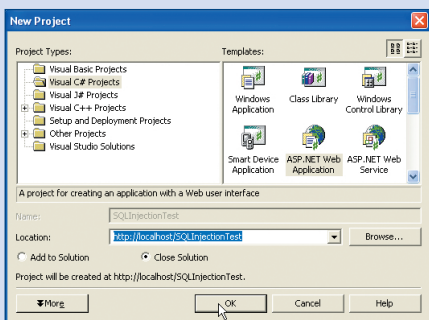
Scopriamo come gli hacker operano per accedere alle informazioni del database e ne modificano il contenuto attraverso semplici web form. Applichiamo le tecniche di difesa per prevenire il SQL-Injection

Uno dei temi che più caratterizza il panorama informatico attuale, riguarda la sicurezza delle applicazioni. Ogni giorno si scoprono nuove falle, bug che un qualsiasi hacker può sfruttare per attaccare il sistema. Nello sviluppo delle applicazioni, molto spesso purtroppo, la sicurezza viene trascurata o messa in secondo piano per diversi motivi, come ad esempio i tempi e i costi che ne derivano per sostenerla. Questo comporta una scarsa affidabilità, mentre basterebbero semplici accorgimenti per evitare grandi catastrofi. È questo il caso degli attacchi di tipo SQL-Injection. Si tratta di una tecnica tramite la quale un utente malintenzionato, inserendo istruzioni SQL in un campo di input, potrebbe ottenere informazioni riservate, entrare a conoscenza dei nomi degli oggetti che compongono il database o addirittura eseguire istruzioni dannose come la cancellazione completa di una tabella. In questo articolo impareremo come un attacco può avere luogo e quali sono i danni da esso procurati, per capire, poi, quali sono gli accorgimenti più indicati per proteggerci. L'ambiente che più si presta a questo tipo di attacco è certamente quello Web, perciò utilizzeremo ASP.NET, C# e SQL Server per i nostri esempi, ma bisogna tener presente che anche le applicazioni desktop non sono del tutto immuni. Le tecniche utilizzate possono essere facilmente applicate anche ad altre tecnologie, linguaggi o database.

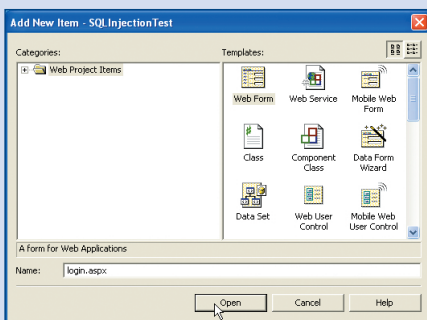
SQL-INJECTION IN AZIONE: LOGIN NON AUTORIZZATO

Creiamo un nuovo progetto ASP.NET C# seguendo i passi riportati nella sequenza in fondo alla pagina, in modo da ottenere un webform simile a quello riportato in **Figura 1**.
Nel gestore dell'evento *click* per il controllo *btnLogin*, raggiungibile con un doppio click sul controllo stesso, inseriamo il seguente codice:

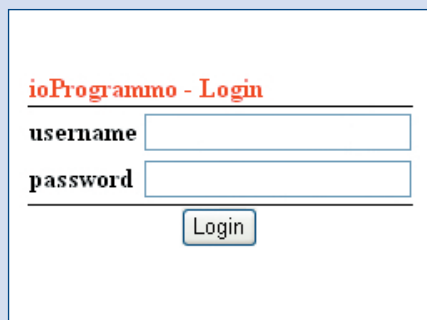
```
string username = txtUsername.Text;
string sql = "SELECT USERNAME FROM USERS WHERE
  USERNAME = '" + username + "' AND PASSWORD = '"
```

IL NOSTRO AMBIENTE DI TEST

CREIAMO IL PROGETTO

1 Con Visual Studio .NET creiamo un nuovo progetto ASP.NET C# chiamandolo *"SQLInjectionTest"*

LA PAGINA DI AUTENTICAZIONE

2 Eliminiamo la pagina *webform1 .aspx*, automaticamente generata, ed aggiungiamo la pagina *login.aspx*

COSTRUIRE IL FORM DI AUTENTICAZIONE

ioProgrammo - Login
username
password
Login

3 Aggiungiamo i diversi controlli in modo da ottenere un risultato simile alla figura riportata.

```
                              + txtPassword.Text + "'";
System.Data.SqlClient.SqlConnection conn =
          new System.Data.SqlClient.SqlConnection();
conn.ConnectionString = @"Integrated Security=
                   SSPI;Data Source= localhost;Initial
                           Catalog=SQLInjectionDb";
System.Data.SqlClient.SqlCommand cmd =
    new System.Data.SqlClient.SqlCommand(sql, conn);
conn.Open();
object result = cmd.ExecuteNonQuery();
conn.Close();
if (result!=null)
    System.Web.Security.FormsAuthentication
        .RedirectFromLoginPage(txtUsername.Text, false);
else
            Response.Write("Utente non autorizzato");
```

Concatenando istruzioni SQL con gli input immessi nel form di autenticazione, viene generata una query che, attraverso il metodo *ExecuteNonQuery*, interroga il database e restituisce il primo campo del primo record che risponde ai vincoli impostati. Se viene trovato almeno un record, si assume che l'utente esiste e viene loggato correttamente, altrimenti il risultato sarà un valore *null* e di conseguenza il login fallisce. Un codice simile è utilizzato molto spesso in questi scenari poiché consente di ottenere un risultato valido ed immediato. Nasconde, però, diverse insidie. Verifichiamo cosa potrebbe accadere se un utente malintenzionato tentasse di attaccare il nostro sistema iniettando codice SQL. Proviamo ad inserire nel form di login i seguenti valori:

- **Username:** *' OR 1=1 --*
- **Password:** *untestoqualsiasi*

Al contrario di quello che potevamo aspettarci, veniamo correttamente autenticati dal sistema, come potete vedere in **Figura 1**. Ma come è possibile? Provando ad analizzare la stringa risultante dalla concatenazione della query con il valore dei campi di testo, otteniamo:

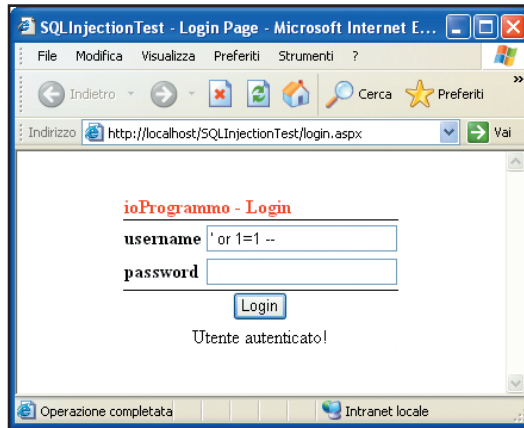



Fig. 1: *Il form di autenticazione della nostra applicazione*

QUATTRO SEMPLICI REGOLE DA SEGUIRE

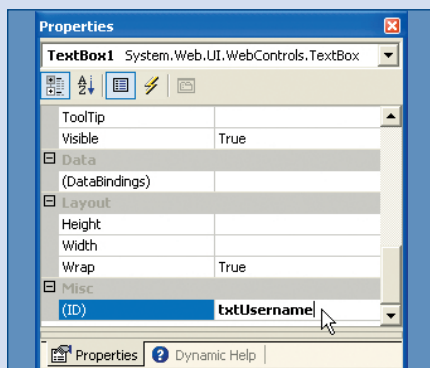
Dopo aver appreso i metodi che un hacker può applicare per attaccare il nostro sistema, possiamo stilare una breve check-list di azioni per evitare danni irreparabili:

1) **Utilizzare query parametriche al posto di una semplice concatenazione di stringhe. Questo permette di rilasciare al database il compito di validare gli input**

2) **Gestire gli errori per evitare che un hacker possa entrare in possesso di informazioni delicate come nomi di tabelle e di campi**

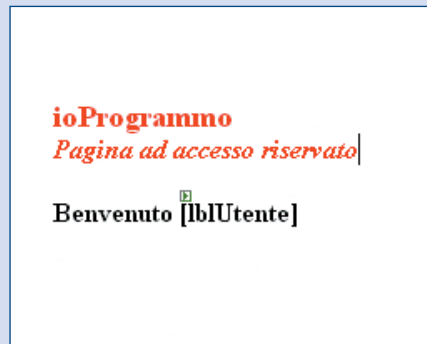
3) **Validare gli input limitando, ad esempio, il numero di caratteri o evitando, dove possibile, l'inserimento di caratteri speciali come gli apici. Il .NET Framework, come molte altre tecnologie, consente l'utilizzo delle Regular Expression, un ottimo mezzo per evitare l'immissione di codice non desiderato**

4) **Utilizzare un account con limitati privilegi di accesso per connettersi al database. In questo modo ad un attaccante è impedita l'esecuzione di codice dannoso come, per esempio, la rimozione di un tabella o l'utilizzo delle stored procedure di sistema come *xp_cmdshell***

IMPOSTIAMO I NOMI

4 **Impostiamo la proprietà *ID* dei controlli textbox con *txtUsername* e *txtPassword* come riportato in figura**

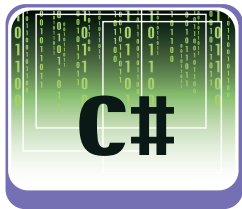
LA PAGINA DI BENVENUTO

ioProgrammo
Pagina ad accesso riservato

Benvenuto [lblUtente]

5 **Creiamo la pagina di benvenuto *default.aspx* che costituirà la nostra area riservata. Ovviamente il contenuto è personalizzabile**

CREIAMO L'AREA RISERVATA

```
Web.config*

<authentication mode="Forms">
    <forms loginUrl="login.aspx"
           timeout="30"
           path="/">
    </forms>
</authentication>
```

6 **Per impostare la pagina *login.aspx* come form di autenticazione, modifichiamo il file *web.config* come visualizzato in figura**

```
SELECT USERNAME FROM USERS WHERE USERNAME
                                    = '' OR 1=1
- - AND PASSWORD = 'untestoqualsiasi'
```

Come potete vedere la query restituisce tutti i record presenti nella tabella poiché l'espressione *"1=1"*, valutata su ogni riga, è sempre vera. Il doppio trattino indica, invece, l'inizio di un commento su linea singola, pertanto le istruzioni seguenti non vengono valutate. Il codice, quindi, ottiene sempre un risultato positivo, consentendo l'accesso anche ad utenti non autorizzati.

VERIFICARE LA NOSTRA APPLICAZIONE

Per essere sicuri che la nostra applicazione sia immune dagli attacchi di SQL-Injection, è necessario testare ogni singolo campo di input. Gli attacchi, infatti, possono aver luogo, oltre che nei form di autenticazione, anche attraverso semplici form di ricerca o di registrazione dati. Partiamo dal presupposto che ogni campo di input può rivelarsi dannoso. Proviamo, quindi, a compilare una check-list di azioni da eseguire per testare gli input della nostra applicazione:

- ✔ **Utilizziamo caratteri speciali che costituiscono caratteri speciali per il linguaggio SQL come gli apici singoli, delimitatori di stringa, i due trattini consecutivi, che indicano l'inizio di un commento su singola riga, o il carattere #, delimitatore di valori di tipo data. In questo caso bisogna bilanciare il controllo. Alcuni input, infatti, possono richiedere l'inserimento di alcuni di questi caratteri. Ad esempio, un campo di inserimento e-mail dovrebbe validare la stringa fabio-cozzolino@ioprogrammo.it e non fabio-cozzolino@ioprogrammo.it--**
- ✔ **Verifichiamo l'inserimento di caratteri UNICODE che spesso vengono trasformati nei corrispondenti caratteri ASCII**
- ✔ **Generare volontariamente errori iniettando stringhe SQL errate per verificare la gestione degli errori**
- ✔ **Verificare anche i valori riportati dai campi di tipo hidden. Un hacker, infatti, potrebbe semplicemente salvare la pagina in locale, modificare il valore del campo hidden ed rieseguire la pagina localmente per tentare l'attacco**

NOTA

ASP.NET permette di configurare l'applicazione per gestire gli errori lato server. Utilizzando l'attributo *defaultRedirect* del tag *customErrors* nel file *web.config*, è possibile indicare una pagina per visualizzare messaggi di errore personalizzati. In questo modo riusciamo a nascondere informazioni che gli utenti non dovrebbero visualizzare.

OTTENERE INFORMAZIONI SUL DATABASE

Una volta scoperto il baco nella pagina, un hacker è in grado di creare seri danni all'applicazione. Utilizzando lo stesso form di autenticazione del primo esempio, proviamo ad inserire i seguenti valori:

- **Username:** *' having 1=1 --*
- **Password:** *untestoqualsiasi*

La parola chiave *having*, che specifica una condizione di ricerca per un gruppo o una funzione di aggregazione, presuppone l'aggiunta della clausola *GROUP BY*. In assenza di questa o di una funzione di aggregazione come potrebbe essere l'istruzione *COUNT(*)*, viene generata una eccezione. E questo è proprio il comportamento che un hacker si aspetta di ottenere. Quando l'applicazione, o i componenti ad essa collegati, generano una eccezione, e questa non è gestita, viene mostrata una pagina che riporta i dettagli dell'errore, fornendo spesso informazioni sulla struttura del database. In questo modo, un utente può facilmente ottenere informazioni sugli oggetti che compongono la base di dati. Partendo da questa considerazione, un hacker può tentare attacchi ancora più dannosi. Come? Purtroppo molti siti web utilizzano l'account di default (*sa*) per connettersi alla fonte dati. Di default, l'account ha la possibilità di eseguire funzioni avanzate sul database come, ad esempio, rimuovere una intera tabella. In **Figura 2** è rappresentato quanto appena detto.

Fig. 2: *L'errore volutamente generato da un hacker*

Se in possesso di queste informazioni, un utente, oltre ad autenticarsi, è in grado di provocare danni irreversibili per il database. Per verificare come l'utente può agire, dopo aver ottenuto le informazioni che ci servono, modifichiamo l'input del form con questi dati:

- **Username:** *' or 1=1; DROP TABLE USERS*
- **Password:** *untestoqualsiasi*

L'istruzione, così composta, comporta l'esecuzione di una seconda query che causa la completa cancellazione della tabella *Users*. La maggior parte dei database consentono l'esecuzione di più query in un'unica istruzione. Sfruttando la stessa tecnica, un hacker può generare qualsiasi istruzione SQL valida oltre a poter eseguire comandi specifici del database sfruttando stored procedure di sistema come *xp_cmdshell*. Tramite questa stored procedure è possibile eseguire comandi di sistema come, ad esempio, eliminare file dall'harddisk o fermare il servizio di SQL Server. Un esempio?
Fate attenzione, però, nell'esecuzione di questo esempio poiché potrebbe risultare dannoso per il sistema se non eseguito come di seguito spiegato. L'autore e ioProgrammo declinano ogni responsabilità. Create un semplice file di testo nella cartella di sistema (es.: *C:\Windows\System32*) chiamandolo *prova.txt*, assicu-

ratevi di averlo creato correttamente e nel form di autenticazione inserite queste istruzioni:

- **Username:** *' or 1=1; exec master.dbo.xp_cmdshell 'del prova.txt' –*
- **Password:** *untestoqualsiasi*

L'esecuzione di questa istruzione porta alla cancellazione del file *prova.txt* precedentemente generato. Come sapete, la cartella di sistema contiene diversi file molto importanti per la corretta esecuzione del sistema operativo e delle applicazioni in esso installate. La rimozione anche di un solo file potrebbe seriamente compromettere la stabilità del server in uso.

CORRIAMO AI RIPARI QUOTING DEGLI INPUT

Ponendo maggiore attenzione nella progettazione e nello sviluppo delle nostre applicazioni, molti dei problemi causati possono essere facilmente evitati utilizzando diverse tecniche. Tra le varie, quella più spesso utilizzata per semplicità ed immediatezza è chiamata *quoting*. Questa tecnica consiste nel sostituire ogni singolo apice con due apici singoli. Modifichiamo, quindi, il codice per l'interrogazione del database inserendo le seguenti righe:

```
string username = txtUsername.Text.Replace("'", "''");
string sql = "SELECT USERNAME FROM USERS WHERE
    USERNAME = '" + username + "' AND PASSWORD =
                    '" + txtPassword.Text + "'";
```

Questa operazione genera una query perfettamente valida poiché i due singoli apici vengono interpretati come un singolo apice parte integrante dello username.
Equivale a dire che stiamo tentando di accedere con una username pari a "' *or 1=1 --*", un valore che ovviamente non esiste nel nostro database. La query non restituisce mai un risultato e l'utente, di conseguenza, non viene autenticato. Purtroppo il *quoting* non è una tecnica sempre applicabile poiché funziona solo in presenza di valori di tipo stringa. Un esempio ci può aiutare a capire il perché.
Ipotizziamo di esserci autenticati. Nella nostra applicazione aggiungiamo una semplice webform con il compito di visualizzare l'elenco degli articoli che contengono una particolare parola e pubblicati in un dato anno. Costruiamo la pagina in maniera similare a quanto visualizzato in **Figura 3**.
Al click sul bottone di ricerca inseriamo il seguente codice:

```
Stored Procedure (SP_TEST)
Name SP_TEST    Owner dbo
CREATE PROCEDURE [dbo].[GETUSERBYUSERNAME](@username varchar(255) output
AS
DECLARE @SqlString varchar(255)
SET @SqlString =
'SELECT * FROM USERS WHERE username = ''' + @username + ''''

EXEC(@SqlString)

RETURN @@ROWCOUNT
GO
```


Fig. 3: ***Il form di ricerca oggetto del nostro attacco***

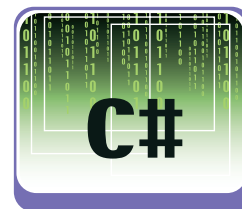
```
// txtChiave corrisponde al campo di ricerca
// txtAnno corrisponde al campo che imposta il filtro
                                          dell'anno
// dgResult è il datagrid che visualizza i risultati
string chiave = txtChiave.Text.Replace("'", "''");
string sql = "SELECT * FROM ARTICLES WHERE TEXT
        LIKE '%" + chiave + "%' AND ANNOPUB =
                              " + txtAnno.Text;

System.Data.SqlClient.SqlConnection conn =
        new System.Data.SqlClient.SqlConnection();
conn.ConnectionString = @"Integrated Security=SSPI;
                  Data Source=localhost;Initial
                  Catalog=SQLInjectionDb";
System.Data.SqlClient.SqlCommand cmd =
    new System.Data.SqlClient.SqlCommand(sql, conn);

conn.Open();
dgResult.DataSource = cmd.ExecuteReader(
    System.Data.CommandBehavior.CloseConnection);
dgResult.DataBind();
conn.Close();
```

Popoliamo il form di ricerca con le seguenti informazioni:

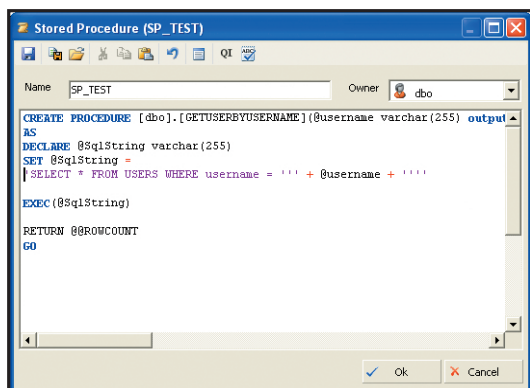
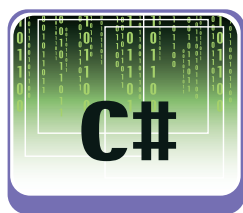
- **Chiave di ricerca:** *sql*
- **Anno:** *2004; SHUTDOWN WITH NOWAIT*

La query risultante dalla concatenazione è la seguente:

```
SELECT * FROM ARTICLES WHERE TEXT LIKE '%sql%'
   AND ANNOPUB = 2004; SHUTDOWN WITH NOWAIT
```

Cioè una query funzionante in grado di interrompere l'esecuzione di SQL Server tramite l'istruzione *SHUTDOWN WITH NOWAIT.*
Questo causerebbe un blocco immediato della nostra applicazione, costringendo l'amministratore a utilizzare *SQL Server Enterprise Manager* o *SQL Server Service Manager* per riattivare il servizio.

NOTA

ADO.NET fornisce quattro metodi per l'esecuzione di query sul database:

- **ExecuteReader: restituisce il risultato della query in un oggetto *DataReader***
- **ExecuteNonQuery: esegue la query sul database e restituisce il numero dei record interessati**
- **ExecuteScalar: esegue la query sul database e restituisce il primo campo del primo record interessato**
- **ExecuteXMLReader: restituisce un oggetto *System.Xml.XmlReader* compilato con i risultati dell'istruzione SQL specificata per *SqlCommand***

REPLACING DEGLI INPUT

Un'altra tecnica spesso utilizzata per prevenire gli attacchi, consiste nel ricercare e rimuovere le parole chiave tipiche che possono essere inserite in un form di ricerca. Il funzionamento è del tutto simile a quello utilizzato per il *quoting*.
Il codice seguente, che precede la concatenazione delle istruzioni SQL, rimuove, ad esempio, le parole chiave che abbiamo finora utilizzato per i nostri esempi, ripulendo la query:

```
string[] keyWords = {"exec", "drop", "shutdown",
                                        "nowait", "--"};
string anno = txtAnno.Text;
for(int i=0;i<keyWords.Length;i++)
{
   anno.Replace(keyWords[i], "");
}
```

Anche se la soluzione può sembrare molto utile, a mio parere il codice si presta poco a variazioni ed aggiornamenti successivi, creando una soluzione potenzialmente complessa da manutenere.

UTILIZZO DI QUERY O DI STORED PROCEDURE PARAMETRICHE

Laddove gli strumenti a disposizione lo permettano, la soluzione ottimale da applicare per evitare gli attacchi è certamente l'utilizzo di query o di stored procedure parametriche.
Questa tecnica relega completamente al database il compito di validare gli input e di generare la query.
Inoltre, l'esecuzione di una query parametrica risulta essere molto più veloce rispetto ad una query concatenata all'interno del codice. Modifichiamo, quindi, il codice contenuto nella pagina di login come di seguito:

```
string sql = "SELECT USERNAME FROM USERS WHERE
         USERNAME = @username AND PASSWORD =
                                        @password";

System.Data.SqlClient.SqlConnection conn =
         new System.Data.SqlClient.SqlConnection();
conn.ConnectionString = @"Integrated Security=SSPI;
                              Data Source=localhost;
                      Initial Catalog=SQLInjectionDb";
System.Data.SqlClient.SqlCommand cmd =
  new System.Data.SqlClient.SqlCommand(sql, conn);

SqlParameter pUserName = new SqlParameter(
              "@username", SqlDbType.VarChar, 30);
pUserName.Value = txtUsername.Text;
cmd.Parameters.Add(pUserName);

SqlParameter pPassword = new SqlParameter(
              "@password", SqlDbType.VarChar, 20);
pPassword.Value = txtPassword.Text;
cmd.Parameters.Add(pPassword);

conn.Open();
object result = cmd.ExecuteScalar();
conn.Close();

if (result!=null) System.Web.Security
        .FormsAuthentication.RedirectFromLoginPage(
                         txtUsername.Text, false);
else
         Response.Write("Utente non autorizzato");
```

Provando a forzare l'autenticazione inserendo codice SQL nei campi di input del form, vediamo che l'applicazione risponde perfettamente evitando l'esecuzione del codice maligno.
Questo perché i valori immessi come parametri vengono valicati e concatenati dal database in un momento diverso e successivo a quello della concatenazione della stringa.
Un maggiore grado di sicurezza si ottiene utilizzando le stored procedure. Ma anche in questo caso la sicurezza assoluta non è garantita. Una stored procedure come quella visualizzata in **Figura 3** riporta lo stesso problema evidenziato nel primo esempio.
L'utilizzo delle stored procedure è strettamente legato al database in uso. Infatti, non tutti i database consentono la gestione delle stored procedure. In MySQL, ad esempio, sono state introdotte solo a partire dalla versione 5.

CONCLUSIONI

In questo articolo abbiamo visto come un hacker può tentare un attacco verso la nostra applicazione sfruttando un codice poco sicuro, aperto all'immissione di stringhe SQL non adeguatamente controllate.
Abbiamo poi visto come sia possibile rendere sicuro e robusto il codice al fine di evitare l'input di istruzioni SQL maligne utilizzando semplici accorgimenti.
Infine abbiamo stilato una breve checklist applicabile a qualsiasi linguaggio o database utilizzato. Per i vostri commenti o altre considerazioni è possibile contattarmi attraverso il forum di ioProgrammo oppure attraverso il mio blog, il cui indirizzo è riportato nel box laterale.
Buon lavoro.

Fabio Cozzolino

SUL WEB

Microsoft Security
http://msdn.microsoft.com/security

Il mio blog
http://blogs.ugidotnet.org/fabioc

BIBLIOGRAFIA

• **WRITING SECURE CODE - SECOND EDITION**
Michael Howard e David LeBlanc
(Microsoft Press)
ISBN 0-7356-1722-8
2003

• **ASP.NET**
Stephen Walther
(Apogeo)
ISBN 88-7303-936-7
2002

Elettronica ed interfacciamento

Un GPS completo. Costruiamolo

Costruiamo un fantastico ricevitore GPS ed impariamo i principi su cui si basa lo scambio dati con un PC e poi programmiamo il tutto!

Come avrete ormai intuito, ci apprestiamo a realizzare un completo ricevitore GPS che ci consentirà di non perdere la rotta in ogni circostanza. Nella prima parte dell'articolo ci dedicheremo all'implementazione del circuito elettrico, mentre nella seconda ci dedicheremo alle specifiche che ci serviranno a realizzare il software di interfacciamento. I meno esperti di elettronica non abbiano paura di tentare l'esperimento. La realizzazione è sufficientemente semplice e con un minimo di pazienza ne verremo a capo. I più esperti troveranno che si tratta di una costruzione semplicissima.

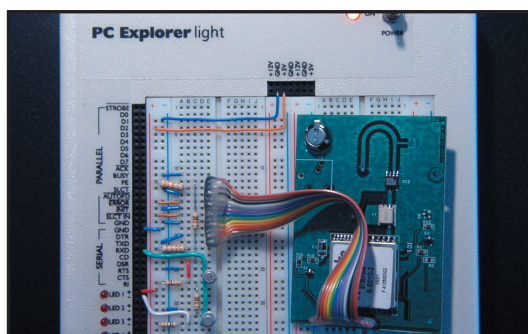



Fig. 1: L'immagine mostra la realizzazione nel suo insieme, completa di modulo GPS ed interfaccia elettronica

IL MODULO GPS

Per rendere la realizzazione dell'applicazione semplice ed intuitiva, oltre che di elevato livello qualitativo è stato utilizzato un modulo GPS dotato di antenna e di sistema di alimentazione integrato prodotto dalla Elisys s.r.l., chiamato *GEOMAG*. Il modulo contiene al suo interno anche una notevole quantità di memoria (8 Megabit), che consente la realizzazione di sistemi di misura e *Data Logging* molto sofisticati. L'interfacciamento del modulo con un qualunque Personal Computer è possibile attraverso la connessione ad un semplice circuito di interfaccia realizzato con l'apparecchiatura PCExplorer light (*www.pcexplorer.it*). Il connettore esterno del modulo comprende una serie di linee che permettono un interfacciamento semplice, affidabile ed immediato. In particolare facendo riferimento a **Figura 4**, si nota che l'alimentazione del modulo può essere attuata ponendo una tensione positiva di +5V sul contatto PACK+, collegato internamente ad un apposito stabilizzatore di tensione, mentre le linee corrispondenti ai Pin *N 1,4,6,8,10* dovranno essere collegate alla massa. Per forzare l'accensione del modulo occorre porre una tensione pari a *+Vcc* sul piedino *GPS_ON*, mentre il contatto *TIMRES#* provvede a resettare il ricevitore ed ad altre particolari condizioni di funzionamento. Il lettore attento avrà senz'altro già notato due porte seriali (*A* e *B*), per mezzo delle quali è possibile realizzare ogni apparecchiatura, anche la più sofisticata, che preveda l'impiego di GPS, comprendendo anche le applicazioni *DGPS (Differential GPS)*. In particolare la porta *'A'* è normalmente utilizzata come canale *free run* secondo il protocollo NMEA, che verrà descritto più avanti, mentre il canale *'B'* per l'accesso alle caratteristiche più intime del modulo, compresa la memoria interna di *DataLogger* di ben 8 Megabit attraverso un protocollo binario proprietario.

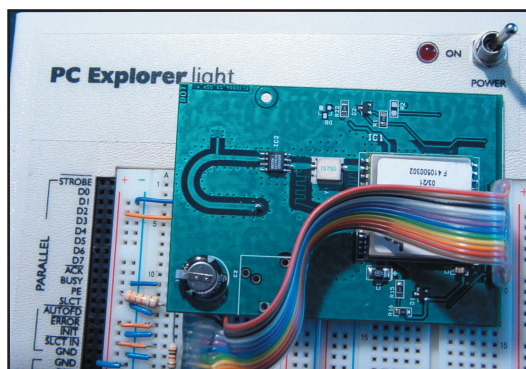
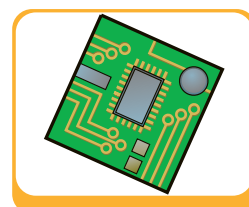
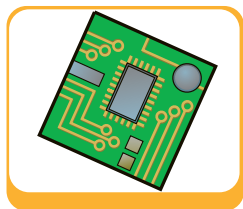



Fig. 2: La figura mostra un dettaglio del modulo GPS GEOMAG (Cortesia Elisys S.R.L.)

Utilizza questo spazio per le tue annotazioni

NOTA

GPS IN AZIONE

Nel CD allegato alla rivista è disponibile un piccolo filmato che mostra la applicazione completa in funzione (GPS_per_PCExplorer.AVI).

PRECAUZIONI

Prima di collegare il circuito al nostro PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato come previsto.

L'utilizzo combinato dei due canali *'A'* e *'B'* può permettere la realizzazione di sistemi GPS differenziali, utilizzando un secondo ricevitore. Nella nostra applicazione utilizzeremo per il momento soltanto il canale *'A'*, sfruttando le enormi potenzialità del protocollo NMEA.

IL CIRCUITO ELETTRONICO DI INTERFACCIA

Facendo riferimento al circuito elettronico seguente ed analizzandolo a partire dal lato destro dello schema, notiamo innanzitutto le connessioni relative al modulo *GPS GEOMAG*. La linea di uscita del segnale seriale sul canale *'A'* del GPS (TXA Pin *N2* del modulo) genera un segnale di una ampiezza massima di 3,3 Volts, compresi tra lo stato logico *LOW* ed *HIGH*, purtroppo non sufficienti ad essere rilevati dalla maggior parte delle porte seriali. Ricordiamo infatti che a seconda del tipo di driver presente in una data porta parallela, i livelli logici possono variare tra +-5 e +-25 Volts, molto superiori alla tensione generata dal modulo GPS. Per ovviare a questo problema è stato inserito a valle del pin *N2* del ricevitore un doppio invertitore (doppio *NOT* logico) costituito dai transistors *TR1-TR2* con lo scopo di convertire il segnale a livelli 0/+5 Volts, sufficienti a pilotare praticamente ogni porta seriale fino alla distanza di qualche metro, distanza adeguata per la nostra applicazione.

Fig. 3: ***L'immagine mostra nel dettaglio le connessioni del modulo GPS GEOMAG (Cortesia Elisys S.R.L.)***

Il risultato di questa conversione è visibile nell'analisi all'oscilloscopio digitale del segnale in uscita dalla linea prima e dopo il gruppo di conversione dei livelli di tensione. Il segnale a questo punto viene inviato alla porta seriale del PC e visualizzato attraverso l'accensione del led *LED1* presente all'interno della apparecchiatura PCExplorer light (*www .pcexplorer.it*). La realizzazione della interfaccia elettronica può essere realizzata con estrema semplicità e senza la necessità di effettuare alcuna saldatura con PCExplorer, che tra le altre caratteristiche comprende anche al suo interno un alimentatore stabilizzato, tutte le connessioni alle porte seriale e parallela del PC oltre a ben dodici diodi LED dotati di resistenza di carico, due deviatori ed un connettore di espansione per ogni applicazione futura di interfacciamento dei Personal Computer.

I pochi componenti elettronici che utilizzeremo per la costruzione del sistema, sono facilmente reperibili in qualunque negozio di componenti elettronici, oppure per corrispondenza presso la Elisys s.r.l. (*www.pcexplorer.it*); l'assemblaggio del circuito può essere effettuato senza saldature per mezzo dell'apparecchiatura PCExplorer light reperibile sullo stesso sito.

Terminata la descrizione della realizzazione Hardware analizziamo il protocollo di comunicazione utilizzato per estrarre le informazioni dal ricevitore GPS.

ESTRARRE LE INFORMAZIONI DAL GPS

Nella maggior parte dei ricevitori GPS è disponibile l'utilizzo di un particolare tipo di protocollo nato per interfacciare dispositivi di navigazione ad utilizzo navale. Questo protocollo, che viene anche implementato nel ricevitore che abbiamo utilizzato nella nostra applicazione, nella versione *NMEA- 0183* è stato definito dalla *'National Marine Electronics Association'* (NMEA). In questa sede faremo riferimento alla versione 2.20.

Il protocollo viene definito come un flusso di stringhe alfanumeriche, i cui campi sono delimitati da virgole e che terminano con un valore esadecimale di *"checksum"* di due caratteri, preceduto da un asterisco. La stringa inizia con *"$"* e termina con "*", tutti i campi vengono trasmessi obbligatoriamente e nessuno di conseguenza è facoltativo. Il calcolo del *"checksum"* avviene operando l'*OR* esclusivo di tutti i caratteri della stringa compresi tra i delimitatori *"$"* e *"*"*, senza considerare questi ultimi. Lo pseudo codice riportato di seguito mostra come il calcolo del *"checksum"* può essere ottenuto in modo semplice e diretto.

Fig. 4: ***Il progetto può essere realizzato seguendo lo schema riportato in figura***

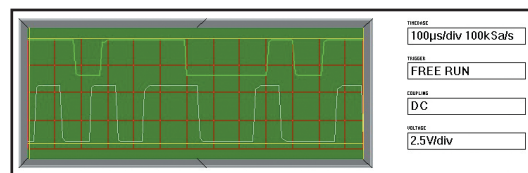



Fig. 5: ***L'analisi all'oscilloscopio mostra la conversione da livelli 3,3V a 5V***

```
Begin
line = getline()
index = 1
checksum = 0
while line[ index] <> '*' do
checksum = checksum EXOR line[ index]
index = index+ 1
end while
End;
```

La sequenza di trasmissione di queste stringhe di informazioni, avviene ciclicamente, con una risoluzione temporale che normalmente raggiunge una frequenza massima di un *'data burst'* al secondo. Il protocollo prevede una innumerevole quantità di *'sentences'* per gli scopi di rilevazione della posizione, ma anche per quanto riguarda la gestione della navigazione fino a giungere alle effemeridi dei satelliti. Il flusso di dati è continuo dal ricevitore verso l'esterno, per mezzo dei canali di uscita del ricevitore che normalmente sono di tipo seriale. L'output che è possibile ottenere da un qualunque tipo di ricevitore che implementa il protocollo NMEA viene riportato, a titolo di esempio di seguito. L'esempio è stato ottenuto utilizzando un comune programma di collegamento e monitor della porta seriale, utilizzato per ricevere le informazioni provenienti dal modulo GPS e dal circuito elettronico descritto di seguito in queste pagine.

```
$GPGLL,4046.908,N,01656.157,E,205503,A*22
$GPBOD,,T,,M,,*47
$GPRMC,205504,A,4046.908,N,01656.157,E,000.0,
                              077,101104, 02,*2B
$GPGGA,205504,4046.908,N,01656.157,E,1,04,4.1,37,*2E
$GPGSV,3,1,11,02,*3E
$GPGSV,3,2,11,10,48,223,*DF
$GPGSV,3,3,11,27,20,069,42,28,05,039,00,*E2
$GPRMC,205505,A,4046.908,N,01656.157,E,000.0,
                              077,101104, 02,*3A
$GPGSA,A,3,,07,08,,,,,,2,*3D
$GPGSV,3,1,11,02,73,132,00,07,11,*34
$GPGSV,3,2,11,10,48,223,00,17,04,308,00,*EA
$GPGSV,3,3,11,27,20,069,42,28,05,039,00,31,03,
                                   034,32,,,,*23
$GPRMC,205506,A,4046.908,N,01656.157,E,000.0,
                           077.0,080601,002.0,*EA
$GPRMB,A,,,,,,,,,,,,V*71
$GPGGA,205506,4046.908,N,01656.1,*23
$GPGSA,A,3,,07,08,,,,,,27,,31,,4.8,4.3,1,*FA
```

Questo insieme di informazioni può apparire incomprensibile, tuttavia è possibile estrarre da questo "stream" di dati anche soltanto i parametri che ci interessano; usualmente posizione geografica ed altitudine, nonostante siano disponibili innumerevoli informazioni quali data ed ora, prua e velocità che possono rilevarsi molto interessanti per innumerevoli applicazioni. Per focalizzare meglio le potenzialità del ricevitore analizziamo brevemente le quattro sequenze di dati più importanti al fine degli scopi che ci siamo prefissati.

LATITUDINE E LONGITUDINE

Estraendo dall'esempio riportato in precedenza la stringa corrispondente a questa 'sentence' otteniamo:

```
$GPGLL,4046.908,N,01656.157,E,205503,A*2
```

In effetti questa frase è probabilmente la più significativa ai fini della definizione della posizione del ricevitore, in quanto fornisce la latitudine geografica, la longitudine e l'orario al quale è stata effettuata la misurazione. Il formato riferito al protocollo *NMEA* per la precisione è il seguente:

```
Latitude ddmm.mmmm
N/S Indicator
Longitude dddmm.mmmm
E/W indicator
UTC Time hhmmss.sss
Status character A A=data valid V=data invalid
Checksum
<CR> <LF>
```

Volendo fare alcune precisazioni, potremmo sicuramente dire che la Latitudine geografica è l'angolo misurato a partire dal centro della terra, tra l'equatore geografico e la posizione del ricevitore ed è misurata in gradi sessagesimali. Analogamente la longitudine è ugualmente un angolo misurato a partire dal centro della terra, ma come proiezione sull'equatore, tra il meridiano di riferimento (che per il geoide *WGS84* corrisponde al meridiano di Greenwich in Gran Bretagna) e la posizione del ricevitore. Questa volta però l'angolo si misura in ore, minuti, secondi, con la convenzione che i 360° del cerchio massimo corrispondente all'equatore corrispondano a 24h. Questa convenzione permette altresì di operare una veloce computazione del fuso orario relativo a due luoghi di longitudine conosciuta.

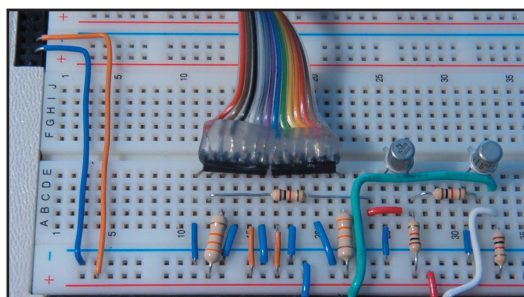
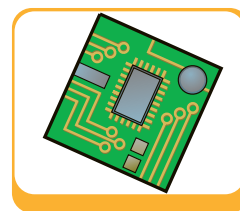



Fig. 6: *Il lettore sarà facilitato nella realizzazione osservando le connessioni riportate nello schema elettrico*

NOTA

ACQUISTARE PC EXPLORER LIGHT

L'apparecchiatura PC Explorer light è prodotta e commercializzata dalla Elisys s.r.l. e può essere acquistata al prezzo di € 213,60 nella versione light, € 99 nella versione basic e € 69 in kit (IVA inclusa) sul web all'indirizzo www.pcexplorer.it oppure inviando una e-mail all'indirizzo pcexplorer@elisys.it, od anche telefonicamente al numero 0823/468565 o via Fax al: 0823/495483.

COMPONENTI ELETTRONICI

REALIZZAZIONE DEL SIMULATORE

N1	PCExplorer light
N1	Modulo Elisys GEOGAP
N2	Transistor BC 107
N2	Resistenza 1Kohm 1/4W
N2	Resistenza 10Kohm 1/4W
N2	Resistenza 33Kohm 1/4W
N1	Condensatore 100 nF

Per la realizzazione è possibile utilizzare PCExplorer light, l'apposito kit di montaggio, oppure una piastra millefori per montaggi sperimentali.

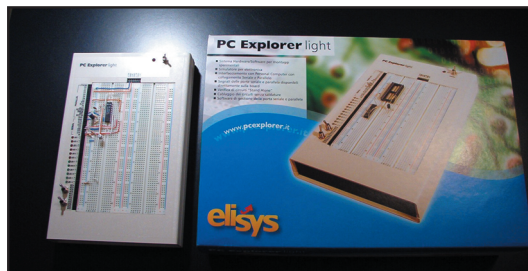
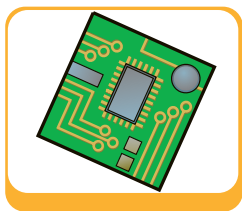



Fig. 7: Per maggiori informazioni sull'apparecchiatura 'PC Explorer light' è possibile visitare il sito: http://www.pcexplorer.it oppure http://web.tiscali.it /spuntosoft/ od infine scrivere all'indirizzo di posta elettronica spuntosoft@tiscali.it

$GPGGA GGA :Global Positioning System Fixed Data

Questa *"sentence"*, introduce oltre alle informazioni di posizione definite già dalla precedente, provvede a fornire alcune indicazioni importanti, qualora siano necessari parametri più raffinati e stime di precisione della posizione più accurate, ad esempio qualora venga utilizzato il GPS differenziale, che permette di raggiungere una precisione elevatissima, utilizzando due ricevitori, uno dei quali fisso che ha lo scopo di rilevare gli errori di posizione di questo sistema combinato. Sottraendo gli errori misurati per mezzo del ricevitore fisso dalla posizione di quello mobile si ottiene una precisione decisamente superiore se paragonata ai sistemi convenzionali dotati di un solo ricevitore. Questo metodo veniva utilizzato anche per annullare gli errori iniettati deliberatamente nel sistema fino a qualche anno fa (la cosiddetta *S.A. Selective Availability*), fortunatamente non più attiva al giorno d'oggi. Dall'esempio estraiamo la stringa *$GPGGA:*

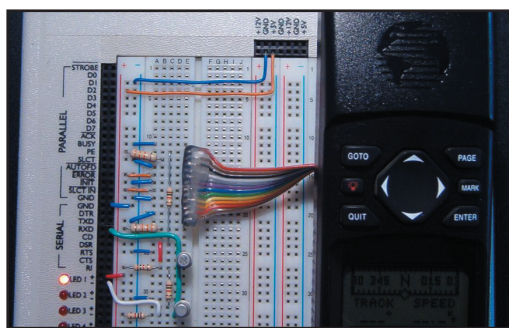



Fig. 8: Per mezzo di questa realizzazione è possibile interfacciare qualunque GPS commerciale

SUL WEB

Il sistema proposto in queste pagine è stato realizzato e collaudato con la apparecchiatura per il collaudo e la sperimentazione di circuiti elettronici con Personal Computer 'PC EXPLORER light': ulteriori informazioni su come si possa reperire questa apparecchiatura è possibile visitare il WEB all'indirizzo: http://www.pcexplorer.it oppure http://web.tiscali.it /spuntosoft/ od infine inviare una e-mail a: spuntosoft@tiscali.it.

```
$GPGGA,205504,4046.908,N,01656.157,E,1,04,4.1,37,*2E
```

Le informazioni disponibili possono essere sintetizzate come:

UTC Time
Latitude ddmm.mmmm
N/S Indicator character
Longitude dddmm.mmmm
E/W indicator character
Position Fix Indicator.
Satellites Used
HDOP numeric (Horizontal Dilution of Precision)
MSL Altitude
Units character M "
Geoid Separation
Units blank M
Age of Differential Corrections Null fields when DGPS is not used
Diff. Reference Station ID
Checksum hexadecimal
<CR> <LF>

Alcune informazioni molto interessanti possono rilevarsi davvero utili per ottenere una stima della precisione anche utilizzando un solo ricevitore GPS. È inoltre disponibile l'altitudine del ricevitore, normalmente espressa in metri.

$GPGSV GSV GPS Satellites in View

Qualora si desideri avere una esatta definizione della situazione dei satelliti che il ricevitore sta 'tracciando' è possibile utilizzare questa frase, spesso frammentata in più 'sentences' consecutive. Per questo motivo il primo parametro che segue l'header è il numero di messaggi che compongono la sentence: nel caso dell'esempio questo numero è corrispondente a tre appunto.

```
$GPGSV,3,1,11,02,73,132,00,07,11,*34
$GPGSV,3,2,11,10,48,223,00,17,04,308,00,*EA
$GPGSV,3,3,11,27,20,069,42,28,05,039,00,31,03,
034,32,,,,*23
```

La struttura della stringa può essere riassunta in questo modo:

Number of Messages
Message Number
Satellites in View
Satellite ID numeric
Elevation numeric
Azimuth numeric
SNR (C/No)
Satellite ID
Elevation
Azimuth
SNR (C/No)
Checksum
<CR> <LF>

In pratica viene fornita la lista dei satelliti ricevuti dal sistema, identificandone il numero, elevazione ed azimut riferiti all'orizzonte ed al nord geografico, oltre che al rapporto Segnale/Rumore misurato.
Questa frase può essere molto utile per monitorizzare lo stato di oscuramento del ricevitore e conseguentemente la bontà delle informazioni ricevute.

$GPRMC RMC : Recommended Minimum Specific GNSS Data

Fino ad ora abbiamo notato soltanto "sentences" ri-

ferite alla posizione geografica del ricevitore ed allo stato dei satelliti che concorrono alla determinazione della posizione. Nel caso si debbano gestire anche i comportamenti cinematici della piattaforma sulla quale è installato il ricevitore, è disponibile questa frase che permette di estrarre, anche prua e velocità riferite al suolo. Probabilmente qualche lettore potrà essere rimasto perplesso da questa ultima frase e si chiederà *perché riferito al suolo? E a che cos'altro?* Per fugare ogni dubbio dobbiamo considerare che il ricevitore può essere montato su qualunque tipo di piattaforma, anche aerea o navale, dove spesso i sensori di velocità e prua effettuano la loro misurazione nei confronti del fluido nel quale sono immersi (l'aria o l'acqua appunto). In effetti analizzando l'estratto del nostro esempio possiamo notare alcune cose interessanti:

```
$GPRMC,205504,A,4046.908,N,01656.157,E,000.0,
                              077,101104, 02,*2B
$GPRMC,205505,A,4046.908,N,01656.157,E,000.0,
                              077,101104, 02,*3A
```

Innanzitutto la frequenza di campionamento è di N1 secondo, confrontando gli orari nei quali è avvenuta la misura. La posizione è identica (Latitudine e Longitudine), confortata dal fatto che siamo fermi e che quindi la nostra velocità è nulla. La prua però riporta 077° rispetto al Nord, come mai se non ci stiamo muovendo? Il sistema memorizza l'ultima direzione misurata, fino a quando non si verifica un nuovo spostamento del ricevitore. In breve i campi possono essere decodificati come segue:

```
UTC Time
Status character A A=data valid or V=data invalid
Latitude ddmm.mmmm
N/S Indicator character N N=north or S=south
Longitude ddmm.mmmm
E/W indicator character W E=east or W=west
Speed Over Ground
Course Over Ground
Date ddmmyy
Magnetic Variation
Checksum
<CR> <LF>
```

Una nota è meritata dall'ultimo parametro prima del *"Checksum"*, ovvero la declinazione magnetica *(Magnetic Variation)*, che misura lo scostamento esistente in ogni punto della superficie terrestre tra il Nord geografico, misurato rispetto alla parallela all'asse terrestre rispetto al meridiano del ricevitore ed il Nord magnetico, ovvero quello che per farla semplice, viene indicato dalla bussola magnetica. Questa differenza esiste dovunque, anche se dalle nostre parti raggiunge al massimo un paio di gradi. L'errore aumenta enormemente alle elevate latitudini ed in particolare in prossimità dei poli e può raggiungere valori di alcune decine di gradi, per cui non è possibile assimilare la prua magnetica e quella geografica. La sentence *"Recommended Minimum Specific GNSS Data"* può essere utilizzata molto convenientemente per la realizzazione di autopiloti, dal momento che ingloba tutte le informazioni che si rendono necessarie per compiere una corretta navigazione tra due punti della superficie terrestre.

IL SOFTWARE

La componente software per la lettura delle stringhe NMEA del modulo GPS attraverso la porta seriale può essere ottenuta dal lettore attingendo ad uno degli innumerevoli programmi di comunicazione seriale disponibili gratuitamente sul Web, come ad esempio gli ottimi *ComDrv16* / *Comdrv32* scaricabili dalla grande rete e completi di codice sorgente. Attraverso qualunque programma di comunicazione seriale, quindi, collegando il circuito di interfaccia che è stato proposto al modulo GPS è possibile attingere al flusso di stringhe NMEA, come mostrato nell'esempio pubblicato. Rimandiamo il lettore al prossimo appuntamento dove svilupperemo insieme una applicazione software completa di gestione di un modulo GPS, che purtroppo non è stato possibile integrare in questo primo articolo, per motivi di spazio sulla rivista ed allo scopo di favorire la qualità e la accuratezza di trattazione dell'argomento.

Fig. 9: **La connessione del GPS al PC avviene per mezzo della porta seriale**

CONCLUSIONI

È stata proposta, in questa sede, l'introduzione ai sistemi GPS ed in particolare all'utilizzo del protocollo NMEA per la realizzazione di un sistema hardware di interfaccia con un ricevitore satellitare. Nel prossimo appuntamento ci rivolgeremo in modo più approfondito alla gestione software di una applicazione rivolta alla determinazione geografica della posizione e della quota di una qualunque piattaforma. Il lettore vorrà comprendere che nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dall'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento.

Luca Spuntoni

L'autore è lieto di rispondere ai quesiti dei lettori sull'interfacciamento dei PC all'indirizzo:
luca.spuntoni@ioprogrammo.it

OpenGis, MySQL e la rappresentazione della traiettoria

GPS e PHP

In questo articolo, parleremo delle estensioni OpenGis di MySQL, diremo qualcosa sulla latidudine e longitudine di un punto, e vedremo come rappresentare una posizione geografica

La geometria spaziale altro non è che quell'insieme di regole che ci permettono l'individuazione nello spazio di un punto o meglio di un oggetto; avrete sicuramente sentito parlare delle coordinate geografiche, o dei sistemi satellitari GPS, ma anche della possibilità di individuare la nostra posizione tramite l'ormai onnipresente telefono cellulare. Tutto questo è geometria spaziale, conosciuta anche col nome *GIS* che significa *"Geographic Information System"*, Sistema di informazione geografica; ovvero l'interpretazione e/o ricerca su una base di dati geografici. In effetti la geometria spaziale può essere tranquillamente applicata anche a dati non strettamente geografici ma che comunque contengano in sè attributi spaziali, si pensi come esempio alle tre coordinate dello spazio, oppure all'andamento nel tempo di un titolo in borsa. Grazie a questo sistema, oggigiorno è possibile non perdersi nelle anguste stradine di montagna. MySQL nella sua ultima revision (4.1) introduce per la prima volta le estensioni *GIS* o "*Spatial Extension*". Come possiamo facilmente intuire i dati geografici non sono facilmente rappresentabili dai classici *"data types"* messi a disposizione da MySQL, per questo motivo sono stati introdotti nuovi tipi che ci aiutano a rappresentare nel miglior modo possibile queste informazioni. Quindi nelle sue estensioni spaziali, MySQL non solo fornisce dei nuovi modelli di dati ma anche una serie di funzioni che ci permettono di interagire con questi dati al fine di poter sfruttare al meglio tutte le potenzialità del *GIS*. A prima vista i tipi di dati che affronteremo, sembrano studiati per rappresentare esclusivamente informazioni geografiche, nella pratica invece si possono astrarre a qualsiasi tipo di informazione anche non strettamente legata a coordinate geografiche, come possono ad esempio essere dei dati che hanno a che fare col tempo e quindi presentano variazioni temporali (es.: trend dei titoli di borsa, andamento meteorologico, ecc.).
Un'informazione spaziale può essere rappresentata principalmente dalle sue coordinate, che, facendo un esempio geografico, possono essere la latitudine, la longitudine e l'altitudine. Il nuovo organigramma della struttura dati, definisce dal più generico dato *"Geometry"* fino ai più specifici e precisi *"Point", "Line"* e così via. Le specifiche includono anche la definizione di due formati "speciali" per la rappresentazione di tali dati all'interno delle queries, ovvero *WKB (Well Known Binary)* e *WKT (Well Known Text).*

I DIVERSI TIPI DI DATI

Lo schema in **Figura 1** illustra in maniera abbastanza chiara e semplice l'organigramma della nuova struttura dati. Definisce dal più generico dato come *"Geometry"* fino ai più specifici e precisi "Point", "Line" e così via. Le specifiche includono anche la definizione di due formati "speciali" per la rappresentazione di tali dati all'interno delle queries, ovvero WKB (Well Known Binary) e WKT (Well Known Text). Una column di tipo *"Geometry"* contiene in sé i seguenti attributi:

- **Il tipo** *(point,curve,surface,ecc.)*
- **Il SRID** ovvero una sorta di ID che viene dato al record istanziato
- **Le coordinate nel sistema di riferimento spaziale**, rappresentate da due numeri a doppia precisione (8 bytes), qualsiasi geometria non vuota includente almeno una coppia di coordinate *(X,Y)*
- **I limiti Interno** *(Interior)*, **Esterno** *(Exterior)* **ed il**

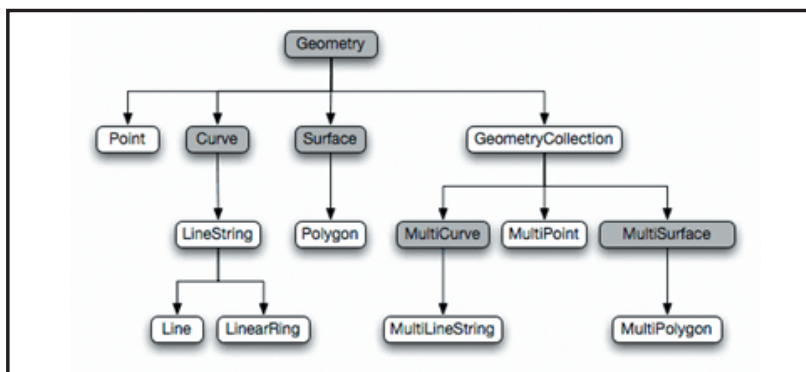
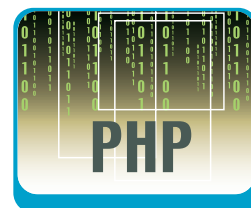
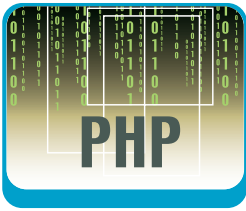



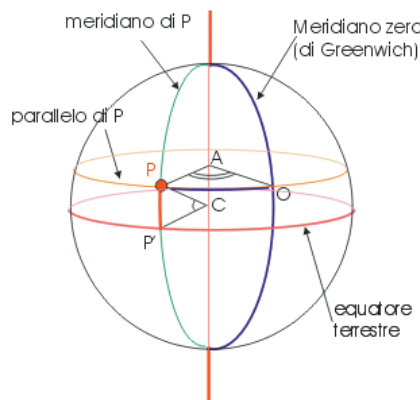
Fig. 1: **L'organigramma della nuova struttura dati utile per definire informazioni di tipo geometrico spaziale.**

confine *(Boundary)*. Ogni geometria occupa una posizione nello spazio. L'esterno è tutto lo spazio non occupato dalla geometria, l'interno è lo spazio occupato mentre il confine comprende l'intersezione tra l'esterno e l'interno.

- **L'MBR** *(Minimum Bounding Rectangle)* o *Contenitore*, questa è l'intera area occupata dalla geometria formata da tutti i minimi e massimi delle due coordinate: (*(minX,minY), (maxX,-minY), (maxX, maxY), (minX,maxY), (minX,-minY)*)
- **La qualità di essere "semplice" oppure "non-semplice", "chiusa" o "non-chiusa", "vuota" o "non-vuota";** ad esempio una geometria si intende vuota quando non ha alcun punto al suo interno, le tre proprietà *Interior,Exterior* e *Boundary* non sono definite. Una geometria vuota è sempre definita "semplice" ed ha un'area pari a 0 (zero).
- **La dimensione** (non inteso in senso di area o grandezza) che può essere:
  - **-1**: geometria vuota
  - **0**: geometria con area nulla e lunghezza nulla
  - **1**: geometria con area nulla e lunghezza non nulla
  - **2**: geometria con area non nulla

In effetti tutti i tipi di dati presenti nella **Figura 1** sono geometrie, essendo tutti astrazioni della classe base *"Geometry"*, ognuno di loro ha però peculiarità particolari e vengono usati in ambiti specifici, facciamo una rapida carrellata:

- **POINT** (rappresenta un singolo punto come ad esempio una singola città in una nazione, oppure un taxi in una città e così via.)
  - Composto dalle coordinate *X* e *Y*

LATITUDINE E LONGITUDINE

Supponiamo che di un certo punto P siano note le coordinate di Latidudine e Longitudine. Queste due coordinate identificano in modo univoco la posizione di P sul globo terrestre. Ma esattamente cosa si intende per latidudine e longitudine? Immaginiamo di sezionare la terra disegnando un piano perpendicolare all'asse terrestre e passante per il punto P. La figura che viene determinata è chiaramente una circonferenza. Consideriamo il punto di intersezione di questa circonferenza con il meridiano di Greenwich e chiamiamo questo punto O. L'interesezione dell'asse terrestre con la circonferenza verrà invece chiamato A. L'angolo formato da PAO corrisponde alla longitudine geografica del punto P. La longitudine può essere Est o Ovest a seconda che il punto si trovi a EST o a OVEST del meridiano di Greenwich e varia da 0 a 180° in senso positivo verso OVEST e negativo verso EST. Per quanto riguarda la latitudine consideriamo invece il piano che passa per P e che contiene l'Asse terrestre. Intersecando questo piano con la sfera terrestre otterremo ancora una volta una circonferenza. Consideriamo il punto C ottenuto dall'intersezione dell'asse terrestre con l'equatore e il punto F ottenuto dall'intersezione della circonferenza di cui sopra con l'equatore. L'angolo PCF indica la latidudine del punto P. Varia da 90° a -90° da NORD A SUD. Va da sé che le due misure di latidutine e longitudine rappresentano facilmente la posizione di un punto sulla sfera terrestre. Da qui in poi vi consiglio di sedervi e leggere con molta calma, perché il paragrafo che segue può essere fonte di mal di testa per i meno amanti della fisica e della matematica. Se pensate di far parte di questa categoria, passate semplicemente a seguire il box con il codice d'esempio di questo stesso articolo. Ora, la nostra idea è molto semplice. Se rappresento su un piano la distanza angolare che il punto P ha da Greenwich e la distanza angolare che P ha dall'equatore, al variare della posizione del punto P riesco a disegnare la sua traiettoria. Sembrerebbe tutto

molto semplice. Ma questa semplificazione non tiene conto che mentre posso considerare costante la dimensione della variazione in senso latitudinale, non posso fare altrettanto per la variazione in senso longitudinale. Infatti in una rappresentazione piana della traiettoria, devo tenere conto che in senso longitudinale la distanza da Greenwich non può essere considerata quella angolare. Essendo difatti la terra sferica, la distanza da Greenwich dipende dalle dimensioni dell'anello che interseca il punto P parallelamente all'Equatore. Perciò nella mia rappresentazione piana non dovrò tenere conto della distanza angolare, ma della lunghezza dell'arco corrispondente agli angoli espressi da latitidune e longitudine. Per fare questo avrò bisogno di conoscere la lunghezza del raggio della circonferenza passante per P ad un dato momento e parallela all'equatore.

ATTENZIONE: se siete sopravvissuti alla lettura di questo paragrafo siete dei veri duri e potete procedere a leggere l'implementazione matematica di quanto detto fino ad ora. Siano

a = 6378,14 il raggio equatoriale terrestre
e = 0,081819822 l'eccentricità del meridiano terrestre

il raggio del parallelo corrispondente alla latidune lat può essere ricavato come segue:

$$R_p = \frac{a\cos(\phi)}{\sqrt{1-e^2\sin^2(\phi)}}$$

dove Φ corrisponde alla latidudine. Una volta ottenuto il raggio della circonferenza, l'arco che sottende l'angolo è banale da recuperare, infatti: la lunghezza dell'arco corrispondente è $R_p\ \Phi$ con Φ espresso in radianti. Per quanto riguarda la determinazione dell'arco di latidudine è sufficiente moltiplicare la latitudine espressa in radianti per il raggio equatoriale terrestre.

- Dimensione: *0*
- Boundary: *empty*

- **LINESTRING (CURVE)** (rappresenta una curva, ovvero una serie di punti interpolati, come ad esempio può essere un fiume oppure una strada)
  - Composto dalle coordinate di una o più coppie di punti (*(X,Y), (X,Y)*)
  - Si definisce *LINE* se è composto esattamente da due punti
  - Si definisce *LinearRing* sè è sia "semplice" che "chiuso"
  - Dimensione: *1*

- **POLYGON (SURFACE)** (rappresenta una superficie, ovvero una serie di curve interconnesse, come ad esempio può essere un intero quartiere, una foresta, ecc.)
  - Delimitato da una serie di *LinearRing*
  - Può contenere dei buchi
  - I *LinearRing* all'interno del poligono non si intersecano

Tutti i tipi fin'ora presentati sono considerati *"Geometrie semplici"*, questo per distinguerli dalle *"Collezioni di Geometrie"* o meglio la classe *GeometryCollection* che permette di definire un insieme di più tipi di base sotto uno stesso oggetto, ad esempio un insieme di *LineString* per indicare una composizione di strade e fiumi, entità separate ma facenti parte di uno stesso "macro-oggetto" come potrebbe essere una città.

QUALCHE ESEMPIO PRATICO

L'esempio che sicuramente ci fa capire bene le potenzialità dell'argomento, è basato sull'uso più classico della geometria spaziale ovvero l'ambito geografico. Immaginiamo di avere a disposizione tutte le coordinate delle strade di una città, così come tutta la situazione dei bus aggiornata in tempo reale in modo da sapere in ogni istante ogni mezzo in quale posizione si trova; creiamo prima di tutto la tabella che contiene i dati della città:

```
CREATE TABLE streets (address CHAR(50) NOT NULL,
  address_geo POINT NOT NULL, PRIMARY KEY(address),
            SPATIAL KEY(address_geo)) type=MyISAM;
```

notiamo da subito l'utilizzo del tipo dati *POINT* a cui andiamo ad assegnare le coordinate della strada *(address_geo)*, una ricerca su una tabella siffatta che potenzialmente conterrà moltissimi dati è impossibile senza l'uso di una adeguata indicizzazione, per cui creeremo un indice strutturato in maniera concreta con la base dati che stiamo utilizzando ovvero un indice *"SPATIAL"*. Popoliamo ora la tabella delle strade con qualche indirizzo di esempio. Ogni riferimento è puramente casuale e le coordinate sono completamente arbitrarie:

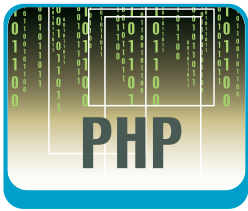
```
INSERT INTO streets VALUES("Via Roma, 10",
               GeomFromText('POINT(2100 2500)'));
INSERT INTO streets VALUES("Via Roma, 14",
               GeomFromText('POINT(2300 2520)'));
INSERT INTO streets VALUES("Via Roma, 18",
               GeomFromText('POINT(2400 2540)'));
INSERT INTO streets VALUES("Via Roma, 22",
               GeomFromText('POINT(2500 2560)'));
INSERT INTO streets VALUES("C.so Europa, 10",
               GeomFromText('POINT(1100 2500)'));
INSERT INTO streets VALUES("C.so Europa, 14",
               GeomFromText('POINT(1300 2520)'));
INSERT INTO streets VALUES("C.so Europa, 18",
               GeomFromText('POINT(1400 2540)'));
INSERT INTO streets VALUES("C.so Europa, 22",
```

La nuova versione di MySQL può essere scaricata direttamente dal sito principale (www.mysql.com), dove in prima pagina svetta l'annuncio della nuova release.
L'indirizzo diretto è http://dev.mysql.com/downloads/mysql/4.1.html da dove potrete scegliere il server in base al vostro sistema operativo. Se usate Windows potete sia scegliere la versione con l'installer automatico (scelta consigliata) sia la versione semplicemente zippata da posizionare e configurare manualmente. Se vorrete usare MySQL 4.1 con PHP seguite le indicazioni fornite con PHP per la configurazione corretta delle librerie client, specialmente nella versione 5 di PHP.

IL FORMATO DATI UTILIZZATO DAL GPS

Tipicamente il formato utilizzato per trasmettere dati da un GPS verso un computer, è l'NMEA. Gestito dal National Marine Electronic Association si applica anche a bluetooth, ai segnali elettrici e all'RS232.
Nel nostro caso ci limiteremo a dire qualcosa sul GPS. Tutte le frasi "NMEA" sono costituite da stringhe ASCII al massimo di 82 caratteri, inclusi i caratteri di controllo. Una frase NMEA inizia con $ e termina con un CR LF, i campi interni sono separati fra loro da una virgola. Immediatamente a seguire al carattere *$* deve essere posizionata una stringa che indica il tipo di periferica che genera la frase. Nel caso dei GPS questa stringa è "GS". Segue il tipo di dato contenuto nella frase. Ad esempio GLL indica Geographic Position Longitutde e Latitude. GLL prevede che i campi che seguono debbano essere così formattati:

1) **1111.11 es. 4512.72** ***Latitudine della posizione attuale***
2) **A es. N** ***emisfero della posizione attuale***
3) **11111.11 es. 000934.88** ***Longitudine della posizione attuale***
4) **A es.** ***E verso della posizione attuale***
5) **hhmmss.ss es. 125642.57** ***ora UTC del GPS***
6) **A es. A** ***stato A=attivo/V=Void***

Infine l'ultimo campo separato da * dagli altri campi è un checksum.
Al momento non ci soffermeremo su come si calcola, ma è semplicemente un codice di controllo della corretta trasmissione della stringa.

```
GeomFromText('POINT(1500 2560)'));
```

In questo contesto utilizzo una funzione molto particolare *"GeomFromText"*, che permette di trasformare nel formato binario gestibile da MySQL *(WKB)* un testo che nel caso specifico rappresenta la coordinata precisa di un punto (che nell'esempio è il punto di riferimento per il numero civico della strada in oggetto) se avessimo voluto indicare un'intera strada (ovviamente più comodo ma molto meno preciso) avremo potuto utilizzare la seguente forma (con l'accortezza di dichiarare il campo *address_geo* come *LINESTRING* anziché *POINT*:

```
INSERT INTO streets VALUES("Via Roma",
GeomFromText('LINESTRING(2100 2500,2300
                    2520,2400 2540,2500 2560)'));
```

Una volta creata la tabella con tutte le strade, istanziamo la tabella che conterrà tutte le informazioni sui nostri bus:

```
CREATE TABLE buses (bus_id INT AUTO_ INCREMENT
      NOT NULL route_number CHAR(5) NOT NULL,
    bus_loc POINT NOT NULL, PRIMARY KEY(bus_id),
             SPATIAL KEY(bus_loc)) type=MyISAM;
```

Come noterete la tabella si presenta in maniera molto simile alla prima (a parte l'id del bus), andiamo ora a popolarla con alcuni dati:

```
INSERT INTO buses VALUES(0, "39",
               GeomFromText('POINT(1990 2000)'));
INSERT INTO buses VALUES(1, "236",
               GeomFromText('POINT(1450 2550)'));
INSERT INTO buses VALUES(2, "236",
```

TRACCIARE LA ROTTA

Supponiamo che si voglia tracciare la rotta che un mezzo di trasporto compie per arrivare da un punto a un altro. Il mezzo potrebbe avere a bordo un GPS che invia i dati in formato NMEA via TCP/ IP a un database MySQL. Arrivata a destinazione, la stringa NMEA viene parserizzata e i dati inseriti in una tabella costruita come segue:

```
$sql=" CREATE TABLE percorso (ID_PUNTO
      bigint(20) NOT NULL auto_increment,
        punto point NOT NULL, nome_punto
   varchar(50) default NULL, PRIMARY KEY
        (ID_PUNTO), SPATIAL KEY (punto)
 ) ENGINE=MyISAM AUTO_INCREMENT=1;
```

La query che inserisce i dati potrebbe essere la seguente

```
$sql="INSERT INTO percorso VALUES(
          '',GeomFromText('POINT($latitudine
            $longitudine)'),'$luogo[name]');";
```

L'unico accorgimento è quello di trasformare i punti latitudine e longitudine inviati dal GPS come segue:

```
latidudine = ore + minuti/60 + secondi/3600
longitudine = ore + minuti/60 + secondi/3600
```

Data la latidudine della Tour Eiffel 48° 51" 32' il dato potrebbe essere trasformato

```
Latitudine= 48+51/60+32/3600 = 49.27246
```

Una volta inseriti i dati il semplice programma d'esempio potrebbe utilizzarli per disegnare una rotta.

CONNETTIAMOCI AL DB

```
$connessione = mysql_connect(
          "localhost", "xxx", "xxx") or die(
              "Connessione non riuscita");
mysql_select_db("geo_db") or die(
    "Selezione del database non riuscita");
$a=6378.14;
$e=0.081819822;
```

1 **Ci siamo connessi al database, inoltre abbiamo settato la costante *$a* pari al raggio equatoriale terrestre e la costante *$e* eccentricità del meridiano terrestre che ci serviranno per calcolare la posizione dei punti.**

PRELEVIAMO LE INFORMAZIONI DAL DB

```
$im = @imagecreate(800, 600) or die(
  "Cannot Initialize new GD image stream");
$background_color = imagecolorallocate(
                    $im, 255, 255, 255);
$text_color = imagecolorallocate(
                      $im, 233, 14, 91);
$sql = "SELECT ID_PUNTO,x(punto) as X,y(
 punto) as Y, nome_punto FROM percorso";
$result=mysql_query($sql);
if (!$result) {
   die('Invalid query: ' . mysql_error());
}
```

2 **Abbiamo inizializzato l'immagine su cui proietteremo la nostra rotta. Inoltre abbiamo recuperato i dati dal DB. Si noti la *SELECT* che sfrutta le funzioni *x()* e *y()* per recuperare i dati dalla collezione di tipo Geometrico definita prima**

TRASFORMIAMO TUTTO IN COORDINATE PIANE

```
while ($row=mysql_fetch_assoc($result)) {
  $latitudine = deg2rad($row['X']);
  $numeratore=$a*cos($latitudine);
  $denominatore=sqrt(1-(pow($e,2)
              *pow(sin($latitudine),2)));
  $rp=($numeratore/$denominatore);
  $longitudine=deg2rad($row['Y']);
  $x=$rp*$longitudine;
  $y=$a*$latitudine;
  $col_ellipse = imagecolorallocate(
                          $im, 0, 0, 0);
  imagefilledellipse($im, 392+($x/10),
        407-($y/20), 8, 8, $col_ellipse);
  imagestring($im, 10, 400+($x/10), 400-
   ($y/20), $row['nome_punto'], $text_color);
}
```

3 **Questa è la parte più importante, le coordinate in termini di latidutine e longitudine vengono trasformate in coordinate piane che ci servono per disegnare l'immagine. Si noti l'uso della funzione deg2rad per trasformare tutto in radianti.**

DISEGNIAMO LA ROTTA

```
header("Content-type: image/png");
imagepng($im);
imagedestroy($im);
```

4 **Le informazioni raccolte al passo 3 vengono raccolte e mandate in output su un'immagine. L'unica nota di rilievo consiste nell'uso di un header per definire il contenuto della pagina.**

```
GeomFromText('POINT(2050 2550)'));
```

Per verificare che abbiamo inserito bene tutti i dati nelle due tabelle, mandiamo in esecuzione una query di ricerca così impostata:

```
SELECT address,asText(address_geo) FROM streets
                              ORDER BY address;
```

la quale ci restituirà l'elenco dei record presenti in tabella nella forma più comprensibile, grazie all'uso della funzione *asText* che traduce la forma binaria dei dati in una forma leggibile testualmente; il risultato della query sarà:

```
+-----------------+----------------------+
| address         | astext(address_geo)  |
+-----------------+----------------------+
| C.so Europa, 10 | POINT(1100 2500)     |
| C.so Europa, 14 | POINT(1300 2520)     |
| C.so Europa, 18 | POINT(1400 2540)     |
| C.so Europa, 22 | POINT(1500 2560)     |
| Via Roma, 10    | POINT(2100 2500)     |
| Via Roma, 14    | POINT(2300 2520)     |
| Via Roma, 18    | POINT(2400 2540)     |
| Via Roma, 22    | POINT(2500 2560)     |
+-----------------+----------------------+

8 rows in set (0.00 sec)
```

Cercando ora di trarre vantaggio da questo tipo di funzionalità, iniziamo ad impostare una ricerca un po' più complessa, che ci permetta ad esempio di cercare il bus più vicino ad una determinata strada:

```
SELECT
 b.bus_id
 b.route_number, ROUND(Glength(LineStringFromWKB(
          LineString(AsBinary(b.bus_loc), AsBinary(
                   a.address_geo))))) as distance
FROM buses b, streets a
WHERE a.address = 'C.so Europa, 20'
ORDER BY distance ASC LIMIT 1;
```

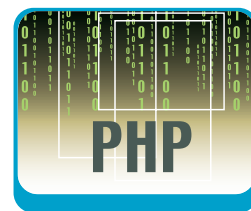
Incredibilmente il risultato sarà:

```
+--------+--------------+-----------+
| bus_id | route_number | distance  |
+--------+--------------+-----------+
|      2 | 236          |        71 |
+--------+--------------+-----------+

1 row in set   (0.00 sec)
```

La query di ricerca a prima vista potrebbe sembrare alquanto complessa ma in effetti non lo è; in pratica chiediamo al db di darci il bus che si trova più vicino all'area della strada *"C.so Europa, 20"*. Questo è possibile grazie alla funzione *Glength* che calcola la lunghezza di un segmento. Se notate bene abbiamo creato all'occorrenza un segmento *(LineString)* composto dalle coordinate del bus: *AsBinary(b .bus _loc)* e dalle coordinate della strada *AsBinary(a.address_ geo);* la funzione *LineStringFromWKB* viene utilizzata per trasformare il segmento in un formato comprensibile alla query, diversamente avremmo dei risultati non coerenti con la ricerca effettuata; alla fine il risultato sarà il bus che è più vicino all'indirizzo cercato con relativa distanza, calcolata tenendo conto del sistema di riferimento utilizzato per l'immissione delle coordinate all'interno del database.

CONCLUSIONI

Le funzionalità del GIS vengono utilizzate appoggiandosi su basi dati che fanno uso di funzioni molto simili a quelle discusse . Gli esempi proposti sono volutamente semplici per poter rientrare nell'ambito didattico-introduttivo di cui l'articolo fa parte. Tutte le peculiarità esposte possono essere sfruttate interfacciando MySQL con i più diffusi linguaggi di programmazione. In particolare PHP soprattutto nella sua ultima release lo rende il compagno ideale per tutte le killer application che si propongono di implementare la localizzazione di informazioni geografiche.

Tommaso D'argenio

GESTIRE LE QUOTAZIONI DI BORSA

Come detto all'inizio dell'articolo, le estensioni spaziali non si prestano solo all'analisi di dati puramente geografici bensì alla gestione di tutti quei dati che presentano caratteristiche "spaziali" o come meglio potremo definire "tridimensionali" dove abbiamo a che fare con più dimensioni sullo stesso tipo di dato; esempio molto banale ma sicuramente che ci fa capire meglio la parola "tridimensionale" è l'andamento dei titoli di borsa, dove ci troviamo nella situazione di dover analizzare il trend di crescita (o non crescita) dei vari titoli; vediamo come potrebbe essere definita una tabella del genere:

```
CREATE TABLE stock (titolo CHAR(5)
   NOT NULL,data DATE NOT NULL,trend
        LINESTRING NOT NULL, PRIMARY
          KEY(titolo),SPATIAL KEY(trend));
```

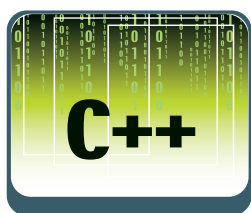
la tabella è molto semplice e contiene il simbolo del titolo (il nome di un qualsiasi titolo di borsa, ad esempio per google avremo *GOOG* e così via), la data di analisi e il trend di quel giorno. Il dato sicuramente più importante è quest'ultimo in cui inseriremo la data ed il valore di apertura e la data ed il valore di chiusura in modo da creare i segmenti; vediamo subito qualche esempio di inserimento dati:

```
INSERT INTO stock VALUES("GOOG",
 "20041123",GeomFromText('LINESTRI
 NG( 20041123 102,20041124 150)'));
```

come vedete abbiamo creato il segmento così come abbiamo creato il punto nell'esempio degli autobus, ovvero utilizzando la comoda funzione *GeomFromText* e la funzione *LINESTRING* con le coppie di coordinate, indicando prima la data (sulla linea delle ascisse) e poi il valore del titolo (sulla linea delle ordinate). In questo modo con dei dati coerenti ed alcune funzioni avanzate di intersezione delle aree (oltre che sfruttando ad esempio le subquery -funzionalità appena introdotta in mySQL) potremo tirar fuori delle analisi molto accurate, utili e soprattutto in tempo reale su dati altrettanto reali (automatizzando il popolamento del database attraverso, ad esempio, alcuni script php) ne più ne meno dei vari servizi di analisi borsistica che sono in linea.

Creiamo da zero un ambiente 3D con Irrlicht

Videogiochi Ambienti 3D

Animazioni, animazioni e ancora animazioni. In questo numero impareremo come "dar vita" ai personaggi dei vostri videogame, facendoli muovere come nella realtà!

Utilizza questo spazio per le tue annotazioni

Nella scorsa puntata abbiamo visto come caricare e visualizzare a schermo un ambiente 3D costruito con gli editor reperibili gratuitamente in Rete. Il risultato è stato di grande effetto e lo sforzo richiesto relativamente poco. La strada per costruire un buon software 3D, tuttavia, è ancora molto impervia e parecchie sono le cose da imparare. Cercheremo di fare in questo articolo un altro piccolo passo in avanti, dando un'occhiata ad alcuni elementi basilari di IrrLicht, come la gestione delle animazioni e degli eventi. Cominciamo!

COS'È UN'ANIMAZIONE?

Per "animazione" intendiamo una qualsiasi deformazione di una mesh 3D che si protrae nel tempo. Per dirla con un esempio: ogni omino-calciatore che vediamo correre su e giù per un prato verde, nel nostro gioco di calcio preferito, è in realtà una mesh animata. Animata perché, per l'appunto, i poligoni che la compongono cambiano forma e posizione col passare del tempo, deformandosi in modo da simulare i più disparati movimenti. Bisogna stare attenti a non confondere "animazione" con "mesh in movimento": il pallone che si muove a seguito dei calci dati dai nostri omini NON è un'animazione. Questo proprio perché non vi è deformazione della mesh che lo rappresenta: rimane sempre una sfera. La definizione di animazione è quindi importante, soprattutto perché IrrLicht gestisce animazioni e movimento in maniera differente.

Per i nostri esempi useremo come modello il file *"sydney.md2"* presente nella cartella "media" dell'installazione di IrrLicht.

Il seguente codice carica il modello da file e lo visualizza a schermo:

```
#include <irrlicht.h>
using namespace irr;
#pragma comment(lib, "Irrlicht.lib")
int main()
{
  // setup dell'ambiente
  IrrlichtDevice* device = createDevice(
                                   video::EDT_OPENGL,
                   core::dimension2d<s32>(640, 480),
                         16, false, false, false, NULL );
  video::IVideoDriver* driver = device->
                                      getVideoDriver();
  scene::ISceneManager* smgr = device->
                                     getSceneManager();
  // carico la mesh
  scene::IAnimatedMeshSceneNode* anms = smgr->
                            addAnimatedMeshSceneNode(
  smgr->getMesh("media/sydney.md2"));
  anms->setMaterialFlag(video::EMF_LIGHTING, false);
  anms->setMaterialTexture(0, driver->
                      getTexture("media/sydney.bmp"));
  // aggiungo la videocamera
  smgr->addCameraSceneNodeFPS(0, 100.0f, 100.0f);
  device->getCursorControl()->setVisible(false);
  // ciclo principale
  while(device->run())
  {
    driver->beginScene(true, true, video::SColor(
                                    255,113,113,133));
    smgr->drawAll();
    driver->endScene();
  }
  // rilascio le risorse
  device->drop();
  return 0;
}
```

L'esempio segue la struttura classica di un programma IrrLicht. In particolare si notino le fasi di setup dell'ambiente, setup della scena, il ciclo principale e il rilascio finale delle risorse utilizzate. Eseguendo questo codice potremo vedere la nostra Sydney che esegue una serie di movimenti apparentemente sconnessi tra loro: corre, spara, cade per terra ecc.
Che succede?

CONTROLLARE L'ANIMAZIONE

Il problema dell'esempio è che IrrLicht non è stato istruito sul come gestire l'animazione di Sydney.
Il file *sydney.md2* contiene, una di seguito all'altra, tutte i movimenti possibili per questa mesh. Le animazioni sono organizzate in fotogrammi (*frame*) ed è comodo pensarle come se fossero una specie di filmato. Un filmato, però, in tre dimensioni.

Fig. 1: *Un esempio di come appare la mesh di Sidney in movimento*

Immaginate di avere a disposizione un video di una persona che cammina per 2 secondi e nei successivi 2 secondi si siede per terra. Manipolando i singoli fotogrammi, potremmo ripetere per un numero indefinito di volte i primi 2 secondi, dando l'impressione di una camminata molto più lunga. Inoltre, mandando al contrario la fase in cui la persona si siede, potremmo "creare dal nulla" una nuova fase in cui compie il movimento opposto: si alza da terra. Montando queste mini-scene in maniera appropriata avremmo la possibilità, in definitiva, realizzare a nostro piacere un filmato, anche molto lungo, nel quale il protagonista cammina, poi si siede, si rialza, cammina di nuovo ecc. In altre parole creeremmo una animazione complessa, montando insieme dei "pezzi" più semplici, in modo da dare l'impressione della continuità dell'azione.

Il discorso per le mesh animate è esattamente lo stesso. Si ha unico "filmato" contenente le singole animazioni di base. Il software ha il compito di scegliere nella giusta maniera, e in tempo reale, l'animazione giusta che risponda, ad esempio, agli input dell'utente o alla situazione di gioco.
Per "fermare" la schizofrenica Sydney possiamo semplicemente aggiungere la seguente riga:

```
anms->setFrameLoop(0, 0);
```

L'istruzione setFrameLoop() imposta la ripetizione ciclica (loop) dell'animazione compresa tra i fotogrammi iniziale e finale, passati come argomento. In questo caso si sta dicendo a IrrLicht di ciclare sempre sullo stesso frame, il primo. Ecco perché, eseguendo ora il codice, sydney apparirà immobile.

3D CAKEWALK

La licenza di IrrLicht permette di utilizzare questo motore anche in progetti commerciali, ottenendo software di notevole impatto, senza sborsare un euro per le royalty. Se questo potrebbe sembrare, per un progetto dopotutto amatoriale come IrrLicht, una specie di "sentiamoci grandi", sappiate che non è così. Esistono diversi software commerciali basati su questo motore. Uno di questi è 3D Cakewalk (http://www.3dcakewalk.com) un software di tipo "clik'n'play" per la creazione veloce di videogiochi. Utile per una prototipazione rapida oppure per passare qualche pomeriggio sfogando la propria creatività di game designer! I prezzi partono da 40$ ma è possibile scaricare gratuitamente una versione di prova che dura 30 giorni.

Per ottenere una plastica corsa dovremo invece inserire la seguente:

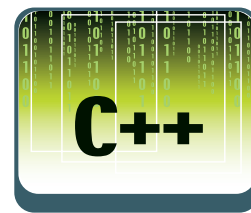
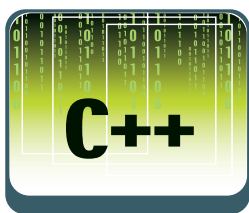
```
anms->setFrameLoop(320, 360);
```

e cioè far "loopare" l'animazione tra i fotogrammi 320 e 360, che contengono appunto il movimento di "corsa". L'utilizzo di questi indici per gestire una animazione potrebbe sembrare scomodo (infatti un po' lo è!) ma è dettato dal fatto che, spesso, i formati dei file 3D non vanno molto oltre questo livello di complessità. Inoltre una gestione così basilare permette di implementare qualsiasi tipo di "gestione personalizzata" da parte del programmatore, ampliandone le possibilità di scelta.
Un altro parametro che permette di controllare le animazioni è la velocità con la quale i frame si succedono a schermo. È possibile variare questa velocità tramite la seguente:

```
anms->setAnimationSpeed(30);
```

più alto è il valore passato come argomento maggiore sarà la rapidità di successione dei frame.

BLENDER

Blender è un editor di oggetti tridimensionali: si tratta di un prodotto professionale e opensource. Lo trovate nel CD allegato a IO Programmo

EVENTI

Il controllo dell'animazione è dunque compito del programmatore. Il più delle volte il giusto alternarsi dei frame deve essere stabilito in base agli input impartiti dall'utente. Ad esempio premendo un tasto il personaggio dovrà correre, premendone un altro dovrà sparare ecc. Sarebbe quindi interessante aggiungere al nostro esempio questo tipo di interattività, che è poi essenziale in un programma 3D real-time. Gli input dell'utente vengono gestiti da IrrLicht con un sistema basato su eventi. Questi eventi sono, ad esempio, input come *"è stato premuto il tasto X"*, *"è stato mosso il mouse"* ecc. Un altro tipo di eventi, che vediamo nel tutorial, sono quelli generati dal sistema di interfaccia grafica (*GUI*) fornito da IrrLicht. In questo caso gli eventi sono del tipo *"è stato premuto un bottone"*, *"è stata incrementata di 10 la scrollbar"* e così via.
Il meccanismo fornito è abbastanza semplice.
I passi da seguire sono essenzialmente due:

1. **Derivazione:** si deriva una classe dalla classe *IEventReceiver* e se ne ridefinisce il metodo *OnEvent()* in modo che gestisca l'input come vogliamo.
2. **Gestore:** si istanzia un oggetto della classe creata al punto precedente e lo si passa come argomento della *createDevice()*, la funzione che istanzia il dispositivo di visualizzazione.

Supponendo di chiamare il nostro gestore di eventi personalizzato *MyEventReceiver*, è sufficiente modificare il codice dell'esempio precedente in modo che la creazione del device avvenga con le seguenti:

```
MyEventReciever receiver;
device = createDevice(video::EDT_OPENGL,
                core::dimension2d<s32>(640, 480),
                    16, false, false, false, &receiver);
```

In pratica viene modificato, rispetto a prima, l'ultimo parametro della funzione.

AUDIOOOO... COME?

I software ludici non sono solamente grafica 3D come se piovesse. Dal punto di vista tecnico molte sono le componenti importanti: la gestione di periferiche di gioco, il codice di rete ecc. Una cosa sicuramente fondamentale è l'audio.
IrrLicht si sposa benissimo con Audiere, una libreria software che offre delle API di alto livello, per riprodurre i più svariati file audio (Ogg Vorbis, MP3, FLAC, WAV ecc.). Audiere, come IrrLicht, è multi-piattaforma e può sfruttare diverse librerie audio di livello più basso, come ad esempio DirectSound per Windows o OSS sotto Linux. La stessa demo che mostra le potenzialità di IrrLicht utilizza Audiere per la riproduzione audio. Audiere è un progetto open-source reperibile al link http://audiere.sourceforge.net/.

GESTIONE PERSONALIZZATA

Il gestore degli eventi deve essere, come detto, un oggetto di una classe derivata da *IEventReciever*. *IEventReciever* presenta un solo metodo pubblico virtuale: *OnEvent()*.
OnEvent() è una funzione richiamata internamente da IrrLicht ogniqualvolta si verifica una qualsiasi delle situazioni che sono riconosciute come "evento". Per questo motivo *OnEvent()* ha un parametro in ingresso, *"event"* di tipo *SEvent*, che serve a capire, nel corpo della funzione, quale sia stato l'evento scatenante della chiamata attuale. Si badi bene: NON siamo noi a dovere passare il parametro *"event"* ma è IrrLicht a farlo, con una chiamata interna. Il nostro compito è quello di gestire l'evento se è di un tipo che ci interessa, altrimenti non dobbiamo fare nulla.
Per questa ragione, tipicamente, il metodo *OnEvent()* presenta al suo interno un costrutto switch

COSTRUIAMO UNA SEMPLICE INTERFACCIA GRAFICA

```
// Imposto la finestra come al solito...
MyEventReceiver receiver;
device->setEventReceiver(&receiver);
device->setWindowCaption(L"Si o No?");
video::IVideoDriver* driver = device->
                              getVideoDriver();
// Ottengo un puntatore all'interfaccia grafica
IGUIEnvironment* env = device->
                        getGUIEnvironment();
```

1 **Creare un'interfaccia grafica con IrrLicht è semplicissimo. Il programma mantiene la solita struttura. Dobbiamo però mantenere un puntatore a IGUIEnvironment.**

AGGIUNGIAMO I PULSANTI

```
// Aggiungiamo due pulsanti
// Il primo parametro contiene le dimensioni
                                   del pulsante
// Il secondo parametro la finestra genitore
// Il terzo l'ID necessario al gestore di eventi
// Il quarto il testo da mostrare
env->addButton(rect<s32>(10,10,630,230
                        ), 0, 101, L"SI");
env->addButton(rect<s32>(10,250,630,470
                        ), 0, 102, L"NO");
```

2 **Il puntatore "env" creato in precedenza consente di aggiungere elementi grafici e funzionali quali pulsanti, scrollbar, caselle di testo ecc.**

IL CICLO PRINCIPALE

```
// Ciclo principale
while(device->run() && driver)
  if (device->isWindowActive())
  { driver->beginScene(true, true,
                    SColor(0,200,200,200));
     env->drawAll();
     driver->endScene();
  }
  //Rilascio delle risorse
device->drop();
```

3 **Anche per un programma che presenta la sola interfaccia grafica, senza scene 3D, la struttura prevista da IrrLicht rimane la medesima. Qui c'è il ciclo principale.**

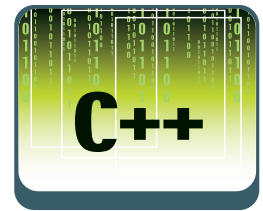
che effettua, appunto, le operazioni specificate, laddove necessarie.
Se quello che vogliamo ottenere dal nostro codice è fare correre *Sydney* tenendo premendo il tasto *SPAZIO*, dovremmo inserire il seguente codice:

```
class MyEventReceiver : public IEventReceiver
{
public:
virtual bool OnEvent(SEvent event)
  {
     if ((anms != 0) && (event.EventType ==
                         irr::EET_KEY_INPUT_EVENT)) {
       switch(event.KeyInput.Key)
       {
         case KEY_SPACE:
         if (event.KeyInput.PressedDown)
           anms->setFrameLoop(320, 360);
         else
           anms->setFrameLoop(0, 0);
         return true;
         break;
       }
     }
     return false;
  }
};
```

La prima cosa che viene fatta nella *OnEvent()* è il controllo delle condizioni iniziali di esecuzione dello switch. In particolare deve essere già esistente il nodo contenente la mesh da animare *(anms)*. Inoltre il tipo di evento deve essere della famiglia dei "tasti premuti" cioè di tipo *irr::EET_KEY_INPUT_EVENT*. Come già detto non tutti gli eventi sono generati dalla pressione di tasti sulla tastiera, per cui questo controllo è d'obbligo. Lo switch poi effettua la sua scelta in base al valore del campo *event.KeyInput.Key*, che è la parte della struttura passata come parametro, che contiene l'informazione sul tasto premuto. Nel nostro caso ci interessa *KEY_SPACE*, la barra spaziatrice. Ci si potrebbe domandare se sia corretto utilizzare uno switch per controllare una sola condizione. In questo caso sicuramente sì. Questa scrittura infatti permette di espandere facilmente il funzionamento del nostro *IEventReciever* con la gestione di altri tasti.
All'interno del *"case"* che ci riguarda, viene effettuato un ulteriore controllo. IrrLicht permette di distinguere tra gli eventi *"tasto premuto"* e *"tasto rilasciato"*.
Questo offre una notevole potenza nella gestione degli eventi da tastiera. Tuttavia potrebbe generare problemi: la pressione dello *SPAZIO*, infatti, scatena in questo modo due eventi (premuto e rilasciato) mentre a noi ne interessa solamente uno alla volta. Per questo imposteremo il loop su *"corsa"* quando *SPAZIO* viene premuto, mentre sarà messo a "fermo" quando *SPAZIO* verrà rilasciato.

CONCLUSIONI

Abbiamo visto in questo articolo alcune delle pedine fondamentali dello sviluppo di una applicazione 3D real-time interattiva. Sono state analizzate le problematiche relative alla gestione delle animazioni collegate a una mesh, nonché il meccanismo di gestione dell'interattività da parte di IrrLicht. Abbiamo inoltre fatto notare come la gestione "basata sugli eventi" sia di notevole generalità è flessibilità. Essa infatti si applica, immutata, sia agli input da tastiera sia a quelli di altro genere, come quelli provenienti dall'interfaccia grafica sviluppata nel tutorial. Nella prossima puntata, che concluderà questo mini-corso su IrrLicht, analizzeremo altre importanti caratteristiche di questo potente motore.
Alla prossima quindi!

Alfredo Marroccelli

Se vi sta piacendo questo motore grafico, ma non siete molto pratici col C++ o semplicemente preferite la comodità offerta dalle miriadi di classi già pronte del linguaggio di casa SUN, non potete non dare un'occhiata a JIRR. Si tratta di un progetto di porting di IrrLicht in ambiente Java, curato da Stefan Dingfelder. Al momento di scrivere siamo a una prematura versione 0.1. Il progetto comunque è disponibile su sourceforge.net quindi chissà che non possiate essere voi stessi a dare una mano a Stefan per portarlo avanti! Il link completo è https://sourceforge.net/projects/jirr/.

GESTIONE DEGLI EVENTI

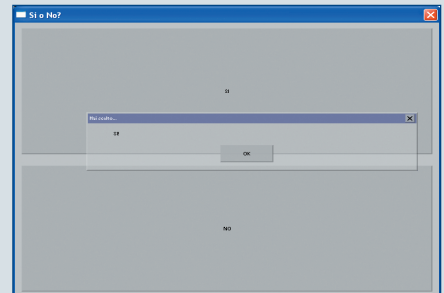
```
virtual bool OnEvent(SEvent event)
{
  if (event.EventType == EET_GUI_EVENT)
  {
     // Ottengo il puntatore
                        all' IGUIEnvironment
     s32 id = event.GUIEvent.Caller->
                                   getID();
     IGUIEnvironment* env = device->
                         getGUIEnvironment();
```

4 **La classe derivata da *IEventReciever* prevede la gestione di eventi generati dall'interfaccia. La condizione iniziale infatti prevede il test col valore *EET_GUI_EVENT*.**

PULSANTI "SI" E "NO"

```
switch(event.GUIEvent.EventType) {
  case EGET_BUTTON_CLICKED:
  // Controllo sugli id dei pulsanti
  // id == 101 --> pulsante "Si"
  if (id == 101) {
    env->addMessageBox(L"Hai scelto...",
                                 L"SI!");
    return true; }
  // id == 102 --> pulsante "No"
  if (id == 102) {
    env->addMessageBox(L"Hai scelto...",
                           L"NO!");  } }
```

5 **Come di consueto dopo la "if" è previsto uno "switch". Questo determina il comportamento vero e proprio da tenere. In questo caso vengono visualizzate delle MessageBox.**

L'INTERFACCIA È PRONTA!

6 **L'aspetto finale del programma è visibile in figura. Ovviamente IrrLicht prevede la presenza di molti elementi grafici, non solo pulsanti, e la possibilità di scegliere skin personalizzate.**

Tecniche di autenticazione con la gestione di certificati X509

Autenticazione e autorizzazione

La sicurezza delle comunicazioni non è una necessità solo nei messaggi scambiati fra persone, ma anche delle transazioni fra macchine. Vediamo qualche applicazione pratica

Utilizza questo spazio per le tue annotazioni

Quando si parla di "autenticazione", sia essa riferita ad un utente o, più in generale, a chi utilizza un servizio, si intende l'identificazione dell'utilizzatore. Può essere necessario autenticare il fruitore di un servizio sia per determinare se esso è autorizzato a fruirne, sia per mantenere traccia del tipo/numero di servizi "consumati". Altre volte è necessario non solo autenticare chi fruisce il servizio, ma anche chi lo eroga: è il caso in cui bisogna essere sicuri che chi eroga il servizio sia chi dichiara di essere e non qualcun altro (per esempio perché altrimenti sarebbe pericoloso fornirgli certe informazioni per noi riservate, quali numeri di carta di credito o coordinate bancarie). In questi casi si parla di mutua autenticazione. La mutua autenticazione può avvenire anche in quei contesti per cui è necessario far sì che i messaggi viaggino in maniera non leggibile da chi non conosce certe informazioni "segrete". Le tecniche per rendere non comprensibili i dati si chiamano tecniche di crittografia.

CRITTOGRAFIA ASIMMETRICA

A partire dagl'anni 70 si è fatta strada una nuova tecnica di crittografia, chiamata "crittografia asimmetrica" o "crittografia pubblica". Essa si basa sul presupposto di utilizzare due chiavi diverse: una utilizzata per cifrare i dati, l'altra per decifrarli. Una delle due chiavi è resa pubblica, una seconda mantenuta privata (e possibilmente segreta). Instaurando la comunicazione con un secondo agente (sia esso un sistema o un'altra persona) gli si fornisce la propria chiave pubblica, mentre lui ci fornirà la sua. Chi invia i dati userà un algoritmo di cifratura che fa uso della propria chiave privata e della chiave pubblica dell'agente a cui invia i dati. Tale agente userà la propria

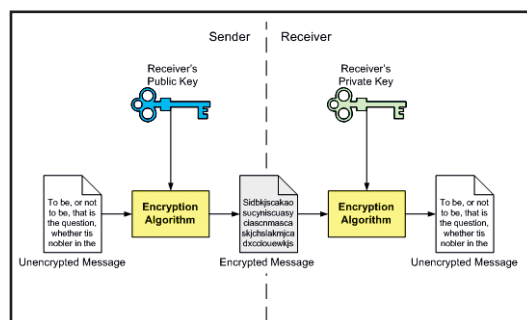



Fig. 1: **Modello per lo scambio di informazioni criptate con il metodo della chiave pubblica**

chiave privata e la chiave pubblica di chi invia i dati per decifrare i dati.
Gli algoritmi di cifratura/decifratura fanno in modo che solo chi è in possesso di *entrambe* le informazioni (quindi almeno una è riservata) sia in grado di spedire/leggere i messaggi. Ovviamente alcuni algoritmi sono più robusti di altri e garantiscono maggiore difficoltà a decifrare i dati se non in possesso delle informazioni necessarie. In linea di principio è possibile provare tutte le possibilità e scardinare ciascun sistema, ma il costo, in termini di tempo e di risorse, è così ampio che nel frattempo è probabile che non ci sia più interesse per il risultato (si pensi al caso in cui è necessario aspettare 100 anni per decifrare una e-mail: probabilmente chi ha interesse a leggerla non sarà più in vita come neppure chi l'ha spedita!). Come si intuisce, questo tipo di crittografia permette anche di garantire l'autenticità delle informazioni scambiate e di autenticare i soggetti coinvolti nella comunicazione e quindi si può raggiungere la "mutua autenticazione" accennata in precedenza.

IL FORMATO X509

Come si evince dal paragrafo precedente, è necessa-

rio ricorrere a formalismi ben definiti per scambiarsi le chiavi pubbliche. Si ricorre a un certificato a chiave pubblica o *"Public Key Certificate"* (abbreviato in PKC). Tra i molti formati di PKC c'è anche il formato *X509* definito in *RFC-2559*. Tra i dati che tale formato specifica possiamo riconoscere quelli relativi al *subject* (dati che identificano coloro i quali detengono la chiave privata: esso può essere una persona, ma anche un'applicazione o un server), chi ha emesso il certificato (issuer), versione del certificato e altri dati a lui relativi (data di scadenza, numero, e così via) ma, soprattutto, contiene la chiave pubblica. Ci sono tre versioni dello standard: l'ultima *(X509v3)* definisce anche delle estensioni. Esse possono essere standard (come l'indirizzo e-mail o DNS, attributi che riguardano il *subject* e l'*issuer* e così via) o non standard (possono includere informazioni specifiche ad un'applicazione o ad un ambito applicativo, come numero di conto corrente, data di nascita della persona, e così via).

SSL E CERTIFICATI

Le applicazioni che probabilmente fanno maggior uso di certificati digitali sono i client Web (browser) per connessioni *SSL* (*Secure Socket Layer*). In *SSL* è sempre necessario che il server sia autenticato. L'utilizzo di *SSL* nella richiesta di pagine Web si evince dal fatto che l'url stessa inizia con *"https"*. Talvolta anche il client possiede un certificato *X509*, ma questo non è vincolante per utilizzare il protocollo: nel caso in cui il client sia sprovvisto di certificato, esso viene autenticato utilizzando username e password, sfruttando la cosiddetta *"basic autentication"* del protocollo *http*. Per realizzare *https* è necessario che il server Web supporti il protocollo *SSL*. Apache è uno di questi ma anche Tomcat può essere configurato per supportarlo. Per poter utilizzare un certificato nelle applicazioni Java è necessario impostare un ambiente per cui:

1. si reperisce un certificato per identificare il server
2. si configura il Web Server per utilizzare *SSL*
3. si associa il certificato al server
4. si reperisce un certificato per il client *(browser)*
5. si associa il certificato al client
6. si usa una Web Application per reperire il certificato presentato dal client (se presente)
7. si usano i dati del certificato presentato

Vediamo, nel dettaglio, come configurare questo ambiente.
Per ottenere un certificato valido si ricorre ad un ente abilitato a rilasciarli; questi enti sono chiamati *Certification Authority* (o semplicemente *CA*). Esistono dei tool per generare certificati di test. Un certificato così generato viene chiamato *self-signed* (ovvero auto-generato, non generato da alcuna *CA*). I certificati self-signed difficilmente saranno utilizzati in un ambiente di produzione, ma sono comodi per iniziare a testare le applicazioni e a conoscere le problematiche connesse.
L'implementazione di Sun dei protocolli sicuri (*SSL* e *TLS*) è il *Java Secure Socket Extensions (JSSE)*. Essa è incorporata nella distribuzione standard a partire dal JDK 1.4. Chi possiede versioni precedenti può scaricarla alla pagina *http://java.sun.com/products/jsse/*. Una volta installato *JSEE* si ha a disposizione anche *keytool*. Questo tool permette di mostrare e trattare (funzioni di import/export) certificati *X.509* in tutte le versioni e di generare certificati *self-signed* di versione 1 usando il comando:

```
keytool –genkey –alias tomcat –keyalg RSA
```

è importante ricordarsi della password impostata, e specificare come *Common Name (CN)* il nome del dominio del server (purtroppo usando keytool esso diventa *"nome cognome"*, che può disorientare l'utilizzatore); se così non è, l'utente che si collega al server verrà avvisato della "anomalia" (anche se tutto potrà funzionare se l'utente accetta di continuare la connessione). Benché sia possibile utilizzare diversi algoritmi nell'opzione keyalg, è preferibile utilizzare RSA per mantenere la compatibilità con diversi componenti che useranno il certificato. Il keystore verrà generato nella home dell'utente che ha eseguito il comando. Se si vuole specificare un file diverso è necessario aggiungere l'opzione:

```
-keystore nome
```

In cui *nome* contiene sia il path sia il nome del file.
In **Figura 2** un esempio di utilizzo per generare un certificato di prova. Un'applicazione Web sviluppata in Java, e quindi composta da Servlet e/o JSP, deve essere installata in un Web Server che la possa eseguire. Tali Web Server si chiamano anche *"servlet container"* (o servlet engine). Jakarta Tomcat (*http://jakarta.apache.org/tomcat*) rappresenta l'implementazione di riferimento da parte di Sun; questo significa che esso deve essere aderente alle specifiche rilasciate da Sun (la versione 5 di Tomcat implementa le specifiche Servlet 2.4 e JSP 2.0). Nel CD allegato trovate la versione 5.5 di Tomcat, altrimenti potete scaricare l'ultima release collegan-

LE VERSIONI DEL FORMATO X509

X.509 Version 1 : disponibile dal 1988; è ancora la più usata, per la sua generalità.

X.509 Version 2 introduce gli identificatori univoci per il subject e l'issuer. Versione poco utilizzata

X.509 Version 3 disponibile dal 1996 e utilizza le estensioni; per questo è personalizzabile.

Fig. 2: ***Uso di keytool per generare un certificato self-signed di prova***

dovi al sito citato. Una volta installato esso risponde (di default) sulla porta 8080.

CONFIGURARE TOMCAT

Tomcat può essere configurato sia per rispondere a tutte le richieste http (modalità *stand-alone*) sia per rispondere solo quando le richieste avvengano verso una applicazione Java.
In quest'ultimo caso è necessario che sia installato un Web Server primario (come Apache, iPlanet o IIS) che "passi" a Tomcat solo le richieste volute.
Questa configurazione è più complessa ma più vicina alle esigenze di performance e sicurezza tipiche degli ambienti di produzione.
I componenti che "colloquiano" o con i browser client o con un Web server (a seconda dell'installazione accennata poc'anzi), si chiamano "connettori".
Essi colloquiano in http, https ma anche in protocolli dedicati, come *AJP* e *WARP*. Per uno stesso Tomcat è possibile configurare più connettori. Per agire sulla configurazione di Tomcat bisogna editare il file *server.xml* presente nella directory *conf/* dell'installazione di Tomcat.
La maggior parte delle opzioni è già disponibile ma in forma commentata, i commenti sono i classici commenti html:

```
<!-- commento-->
```

Per far accettare a Tomcat la connessione *HTTPS* basterà togliere i commenti al connettore *SSL Coyote HTTP/1.1:*

```
<Connector port="8443" maxThreads="150"
        minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" disableUploadTimeout="true"
        acceptCount="100" debug="0" scheme="https"
            secure="true"><Factory clientAuth="false"
                                    protocol="TLS" />
</Connector>
```

si noti che l'attributo *scheme* contiene il nome del protocollo e l'attributo *secure* vale *true*. La porta su cui risponde è specificata in port dell'elemento connector (di default è la *8443*). Si noti che deve essere presente l'elemento *<Factory>*. Se in esso si imposta l'attributo *clientAuth* a *true*, il client deve possedere un certificato per poter instaurare una connessione.
Per installare un keystore che non sia sulla directory home dell'utente, bisogna specificare anche l'attributo *keystoreFile*.
Se, come auspicabile, la password usata per creare il *keystore* non è *"changeit"* (usata di default) si deve specificare anch'essa nell'attributo *keystorePass*. Supponendo che il keystore sia *c:\ioprogrammo* e la password *"ioprogrammo"*, la configurazione del *factory* diventa:

```
<Factory clientAuth="false" protocol="TLS"
      keystoreFile="c:\ioprogrammo" keystorePass=
                                    "ioprogrammo" />
```

Facendo ripartire Tomcat sarà possibile accedervi anche via *SSL*. Si provi l'indirizzo *https://nomehost:8443*. Essendo un certificato self-signed, il browser chiederà all'utente se vuole continuare anche se il certificato non è emesso da una CA attendibile. Sarà anche possibile visualizzare il certificato del server (**Figura 3**).

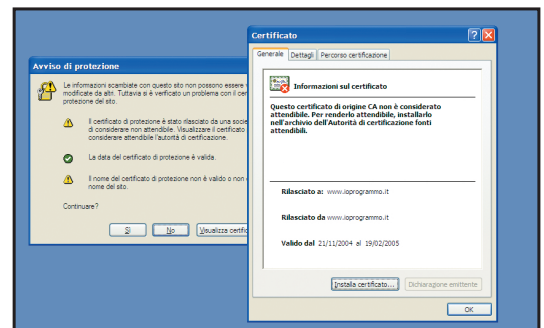



***Fig. 3:** Il certificato del server, visualizzato dal client che ha richiesto la connessione HTTPS*

CONFIGURARE IL BROWSER

Per utilizzare un certificato *X.509* lato client, una volta ottenutolo da una CA, è necessario importarlo all'interno del proprio browser. Se si usa Internet Explorer è necessario accedere al menu *"Strumenti > Opzioni internet…"*, poi andare sul tab *"Contenuto"* e selezionare *"Certificati"*. Appare una finestra come mostrato in **Figura 4**! Per importarne uno nuovo è necessario premere sul tasto *"Importa"*, selezionare il file ove risiede il certificato e procedere alla sua importazione. Esso apparirà sul tab *"Personale"*. In altri tab, come in *"Autorità di certificazione intermedie"*, appaiono quelle CA ritenute affidabili.

CERTIFICATI X509 JAVA

Riuscire a gestire certificati *X509* diventa essenziale

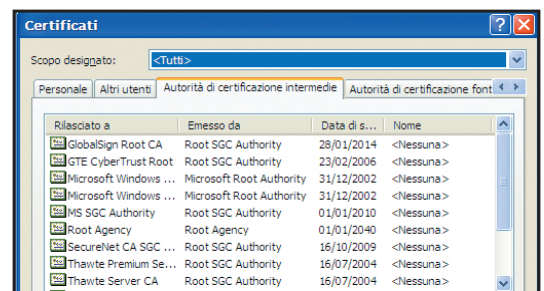



***Fig. 4:** CA "affidabili" in Internet Explorer*

ALCUNI DATI CONTENUTI DA UN CERTIFICATO X509:

- **Version: Versione delle standard X509 usato dal certificato.**
- **Serial Number: Numero che distingue il certificato; usato anche nelle CRL quando scade**
- **Signature Algorithm Identifier: Identifica l'algoritmo usato**
- **Issuer Name: Il nome (secondo lo standard X.500) di chi ha generato il certificato**
- **Validity Period: Periodo di validità del certificato**
- **Subject Name: Nome di chi possiede il certificato. È in formato X500 e si assume unico (distinguibile) in Internet (è composto da varie parti: CN, OU, O, C…)**
- **Subject Public Key Information: La chiave pubblica, utilizzabile per instaurare connessioni a chiave asimmetrica.**

nelle applicazioni che vogliono ricorrere alla crittografia asimmetrica con la mutua autenticazione. In Java esiste il package *java.security.cert* che permette una gestione completa di diversi tipi di certificati. La classe Java che permette di gestire certificati *X509* è *java.security.cert.X509Certificate*. Una Servlet può reperire un eventuale certificato presentato dal client richiedendo l'attributo omonimo dall'oggetto *request:*

```
request.getAttribute("javax.servlet.request.X509Certificate");
```

l'oggetto *request* è l'oggetto che contiene tutti i dati relativi alla richiesta http inoltrata dal client; la servlet lo gestisce attraverso i metodi *doPost* e *doGet* che, rispettivamente, rispondono alle richieste *http(s)* di tipo *"post"* e *"get"*. Ecco una semplice servlet che mostra i certificati presentati dal client (se presenti):

```
public class mostraX509 extends HttpServlet {
  public void doGet(HttpServletRequest req,
   HttpServletResponse res) {
    try{
    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    out.println("Ricerca e stampa di certificati X509");
    if (req.getScheme().equals("https")) {
      X509Certificate[] cert = (X509Certificate[])
      req.getAttribute(
                    "javax.servlet.request.X509Certificate");
      if (cert!=null) {
        for (int i = 0; i < cert.length; i++) {
          out.println("Certificato #"+i+"="+cert[i]); }
      } else {
        out.println("Nessun certificate presente!"); }
    } else {
      out.println("Non è stato usato HTTPS!");}
    }catch(Exception ex) { ex.printStackTrace(); } }
}
```

```
testHttps
  WEB-INF
    web.xml
    classes
      mostraX509.class
```

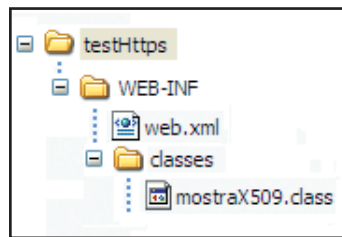



Fig. 5: *La struttura delle directory dell'applicazione*

Questa servlet va compilata e installata, come Web Application, sotto Tomcat. A questo proposito ricordo che è necessario creare una struttura minima sul file system, corrispondente a una directory che contiene la directory *WEB-INF* e al cui interno trovano posto il file *web.xml* che descrive l'applicazione e la *directory classes* che contiene le classi compilate. Ecco come può essere scritto un file *web .xml* minimale: così com'è definito impone che sia la nostra servlet a rispondere per tutti gli indirizzi gerarchicamente successivi a quelli della nostra applicazione:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems,
 Inc.//DTD Web Application 2.2//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
  <display-name>
   Applicazione di esempio
  </display-name>
  <description>Test</description>
  <servlet>
    <servlet-name>mostraX509</servlet-name>
    <display-name>mostraX509</display-name>
    <description>descrizione</description>
    <servlet-class>mostraX509</servlet-class>
    <load-on-startup></load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>mostraX509</servlet-name>
    <url-pattern>*</url-pattern>
  </servlet-mapping>
</web-app>
```

```
Indirizzo https://localhost:8443/testHttps/
Ricerca e stampa di certificati X509
Certificato #0=[
[
  Version: V3
  Subject: OID.2.5.4.13="C=VENUTI/N=IVAN/D=29-03-1972/R=Nessun ruolo", EMAILADDRESS=
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  SunJSSE RSA public key:
  public exponent:
    010001
  modulus:
    a35767d9 296d2d08 f514519f d619aa58 928e7431 c99bf0cf 8f58204d 278e5513
    eb32ad7b 8d301a95 d49bd831 0a7262e5 cb58c88a 6cf98f12 0180404e 374523b3
    c8ad3381 dcb42c8e ac4034d7 3a1aea07 7fbb6870 d5de76aa 6c86035b 005e43b1
    0b97ba92 ef121686 0384f8cd cf36a27d f4e0cfd0 135999ce 5ef7a955 94c8ece9
  Validity: [From: Tue May 20 11:26:57 CEST 2003,
               To: Fri May 19 11:26:57 CEST 2006]
  Issuer: EMAILADDRESS=ca@ione.it, CN='SSL ", O=Rete Telematica , C=IT
  SerialNumber: [    0c52]

Certificate Extensions: 4
[1]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: EB 17 31 0B D6 4B 23 03   60 38 A0 57 E8 C8 0C 12  ..1..K#.`8.W....
0010: EA A9 E4 BC                                         ....
]
]

[2]: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 81 7C 19 07 CA 47 CD 96   60 94 43 C0 AB 40 68 75  .....G..`.C..@hu
0010: 14 8A 58 D4                                         ..X.
```

Fig. 6: *Esempio di funzionamento della servlet che reperisce i certificati X509 presentati dal Client*

In **Figura 6** un esempio del suo funzionamento. Ovviamente è possibile estrarre tutte le informazioni presenti nel certificato. Ecco, per esempio, come estrarre il nome univoco (*DN, distinguish name*) del subject e il suo indirizzo di email:

```
String subjectDN = certs[0].getSubjectDN().getName();
String emailAddress = getEmailAddressFromDN(subjectDN);
```

In Java i certificati *X509* sono utilizzati anche per firmare applicazioni Java, in particolare per firmare file compressi JAR e Applet, con il tool *jarsigner (http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html#security)*. In un prossimo articolo vedremo come poter far uso dei certificati per reperire particolari risorse presenti sul server.

Ivan Venuti

LE CLASSI PRINCIPALI DI JAVA. SECURITY.CERT

- **CertificateFactory** - permette di generare oggetti contenenti certificati e liste di revoca (certificate revocation list CRL)
- **Certificate** - classe astratta che definisce le proprietà comuni di diversi tipi di certificati (funzionalità di codifica e verifica, chiave pubblica e così via)
- **CRL** - classe astratta per gestire *Certificate Revocation List* di diversi tipologie
- **X509Certificate** - classe (astratta) per certificati X509. Definisce gli attribute standard
- **X509Extension** - interfaccia per gestire le estensioni X509 versione 3
- **X509CRL** - classe astratta per gestire *X509 Certificate Revocation List*
- **X509CRLEntry** - classe astratta per modellare elementi di una CRL di tipo X509.

BIBLIOGRAFIA

- **JAVA SERVLET PROGRAMMING** Seconda Edizione *J. Hunter* (O'Reilly) 2001

Le classi Diagnostics per trovare anomalie nell'applicazione

Come ti scovo il bug!

Scopriremo come tracciare il comportamento di un'applicazione, scriveremo un servizio Windows, parleremo di Thread e infine faremo qualcosa di veramente utile!

Utilizza questo spazio per le tue annotazioni

Il debugging e il tracing di un'applicazione sono due tecniche fondamentali per rilasciare un prodotto il più possibile privo di errori ed anomalie. Il debugging permette di analizzare il flusso del codice del programma monitorando valori di variabili e strutture dati in fase di sviluppo. Il tracing, invece, permette di inviare informazioni ad un programma chiamato listener che si mette in ascolto e tiene traccia di eventi eseguiti dall'applicazione durante la sua esecuzione. La sostanziale differenza tra le due tecniche sta nella fase temporale in cui vengono usate. Il Debug è una tecnica che solitamente si utilizza durante la fase di programmazione dell'applicazione, quando ancora si deve rilasciare il prodotto finale. Il Tracing, invece, avviene mentre l'applicazione è stata diffusa ed installata e, soprattutto, è in esecuzione. Utilizzando una sorta di interruttore, tecnicamente chiamato switch, possiamo attivare e disattivare le informazioni che il programma traccia all'interno del listener. Infine, analizzando le informazioni scritte dal programma all'interno del listenester, si possono trovare tracce utili a capire il perchè l'applicazione si sia comportata in un certo modo invece che un altro. L'abilità dello sviluppatore sta nel posizionare le informazioni di tracing in punti del codice che possano essere utili anche dopo il rilascio della versione finale dell'applicazione. Esistono anche casi in cui il tracing risulta l'unica tecnica utile per analizzare il comportamento della nostra applicazione. Pensiamo ad esempio ad un programma che utilizza dei Thread per realizzare più compiti contemporaneamente. Il Debug ci permetterebbe di analizzarne solo uno alla volta rallentando e molte volte, anche, desincronizzando i Thread.

LE CLASSI .NET PER IL TRACING

Le classi fornite da .NET per utilizzare la tecnica del tracing sono tutte contenute all'interno del namespace chiamato *Diagnostics* che deve, quindi, essere necessariamente incluso all'interno del codice che si vuole tracciare. Il *namespace Diagnostics* fornisce quattro classi:

1. **Trace:** questa classe fornisce molti metodi per scrivere messaggi all'interno del listener. Come comportamento standard la classe invia i messaggi alla finestra *Output* di Visual Studio .NET ma utilizzando la collezione *Listeners* è possibile cambiare facilmente questo tipo di azione.
2. **Debug:** è molto simile alla classe Trace, presenta gli stessi metodi e invia informazioni al listener. Differisce per un aspetto fondamentale. Quando si compila in Release per rilasciare il prodotto finale, ogni chiamata a questa classe all'interno del codice viene rimossa dal compilatore.
3. **BooleanSwitch:** per-

COS'È UN THREAD

Il sistema operativo associa un processo per ogni applicazione mandata in esecuzione. Ad ogni processo presente in memoria il sistema operativo riserva una zona di memoria dove allocare strutture dati ed eseguire istruzioni. Inoltre il sistema operativo assegna dei livelli di importanza ad ogni processo permettendo a quelli di sistema di avere una priorità di esecuzione rispetto agli altri. Ogni processo può a sua volta eseguire dei sotto processi, chiamati Thread, che si occupano di eseguire piccole funzionalità parallelamente al processo principale (chiamato anche Thread primario o principale). Per fare un esempio pratico basti pensare a Microsoft Word che, una volta mandato in esecuzione, esegue a sua volta dei Thread per eseguire piccoli compiti come, ad esempio, il controllo ortografico di ciò che si scrive.

mette di definire una sorta di interruttore booleano che attiva/disattiva la funzionalità del tracing.

4. **TraceSwitch:** consente di filtrare i messaggi da inviare al listener basandosi su dei livelli di importanza. Ad esempio, si può scegliere di tracciare tutti i messaggi oppure solo gli errori.

Tra i metodi più utili ed importanti della classe *Trace* possiamo elencare i seguenti:

- **WriteLine:** scrive una riga all'interno del listener.
- **WriteLineIf:** scrive una riga all'interno del listener se la condizione specificata è vera.
- **Flush:** scarica il buffer delle informazioni raccolte verso l'applicazione in ascolto permettendo di scrivere fisicamente all'interno del listener.
- **Close:** chiude le comunicazioni con il programma listener.

Tra le proprietà più importanti va citata, sicuramente, *Listeners* che fornisce una collezione dove aggiungere riferimenti ad applicazioni in grado di ascoltare e tracciare i messaggi inviati dalla classe *Trace*.

LE APPLICAZIONI LISTENERS

Come accennato precedentemente nell'articolo, tutte le istruzioni della classe *Trace* vengono inviate automaticamente alla finestra *Output* di Visual Studio .NET.

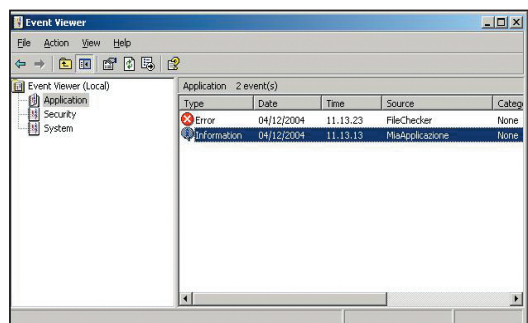
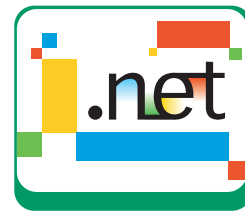



Fig. 1: *L'event Viewer di Windows è un ottimo listener*

Questo comportamento non è certo accetabile quando abbiamo già installato e distribuito la nostra applicazione su altri computer dove non è presente l'ambiente di sviluppo. Per questo motivo possiamo reindirizzare le informazioni inviate dalla classe *Trace* ad un'altra applicazione, utilizzando la proprietà *Listeners*. Questa mantiene una collezione di applicazioni che possono essere messe in ascolto per tracciare le informazioni inviate dal nostro programma.
Tramite il metodo *Add()* possiamo aggiungere un riferimento ad un'applicazione *Listeners* mentre con il metodo *RemoveAt()* possiamo andare a rimuoverne una dalla collezione, alla posizione specificata.

EventLogTraceListener

La prima classe che andiamo ad analizzare permette di dichiarare l'*Event Log* di Windows come applicazione *Listeners*. Il log di eventi di Windows è un sistema che tiene traccia di tre tipologie di eventi: di sistema, di sicurezza e applicativi. Quando utilizziamo questo *Listener* sarà possibile leggere i messaggi inviati dall'applicazione all'interno della categoria *Application*. Vediamo come utilizzare questo Listener:

```
// Creo il nuovo Listener
EventLogTraceListener el= new
                EventLogTraceListener("MiaApplicazione");
// Aggiungo l'applicazione alla collezione
Trace.Listeners.Add(el);
// Scrivo una riga all'interno del Listener
Trace.Write("Entrato nel loop di MioMetodo()"
);
```

La prima istruzione crea la classe *Listener* dedicata alla scrittura di informazioni dentro l'*Event Log* di Windows. Successivamente utilizzando la collezione *Listeners* si aggiunge il nuovo oggetto in modo che venga utilizzato dalla classe *Trace*. Infine chiamando il metodo *Write()* scriviamo un

Fig. 2: *Il dettaglio dell'errore visualizzato dall'Event Viewer di Windows*

COMPILARE L'ESEMPIO

Da Visual Studio utilizzando il combo presente nella toolbar selezioniamo "Release". E compiliamo il tutto usando il menu "Build/Build Solution".

IL METODO MONITORIZZA

Se avete seguito l'esempio proposto in questo articolo, questo codice va inserito nella classe "ImportData". Inizialmente viene creato un oggetto FileSystemWatcher che serve per definire un sistema che si mette in attesa che succeda qualcosa all'interno dellla directory specificata. Ad esempio, possiamo monitorare e scatenare un evento nel caso in cui un file viene creato, cancellato, modificato, ecc. Il costruttore della classe accetta, inoltre, l'estensione del file da controllare. Successivamente l'oggetto viene configurato per scatenare un evento quando viene creato un file XML all'interno della directory monitorata. L'evento Created viene gestito dalla funzione OnCreated() che non fa altro che chiamare la funzione Import() che si occupa di leggere i dati dal file XML, il cui nome è ricavabile dalla proprietà FullPath dell'oggetto FileSystemEventArgs, e importare i dati contenuti all'interno del database.

```
private void OnCreated(object o, FileSystemEventArgs e)
{ Trace.WriteLineIf(m_bs.Enabled,
                    DateTime.Now.ToShortDateString() +
                      ": Iniziata l'importazione dei dati");
  Import(e.FullPath); }
private void Import(string strFileName)
{  try
  {  OleDbConnection cn = new OleDbConnection
       (@"Data Source="C:\Esempio\esempio.mdb"
           + ";Provider=""Microsoft.Jet.OLEDB.4.0""");
     OleDbDataAdapter da = new OleDbDataAdapter(
   "SELECT Au_ID, Author, YearBorn FROM Authors", cn);
     DataSet ds = new DataSet();
     da.Fill(ds);
     DataSet dsMerge = new DataSet();
     dsMerge.ReadXml(strFileName);
     OleDbCommandBuilder cb = new
                         OleDbCommandBuilder(da);
     da.Update(dsMerge);
     Trace.WriteLineIf(m_bs.Enabled, DateTime
          .Now.ToShortDateString() + ": Importazione
                         terminata con successo.");}
     catch(Exception ex)
     { Trace.WriteLineIf(m_bs.Enabled, DateTime
        .Now.ToShortDateString() + ": " + ex.Message);}
  }
}
```

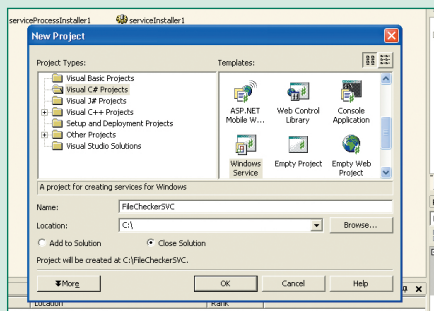
Il codice sembra privo di errori ma, in realtà, esiste un caso che genera un errore che non permette l'importazione dei dati. Quando il file copiato all'interno della directory è molto grande il sistema operativo impiega qualche secondo a terminare l'operazione di copia ma, nel frattempo, il thread scatena l'evento Created che da inizio al processo di import dei dati. Non essendo terminata la copia del file si ottiene l'errore "Il processo non può accedere al file <nome file> perchè attualmente bloccato da un altro processo". Per ricreare questo bug copiate il file import_grande.xml all'intero della directory. Non è detto che si verifichi su tutti i computer, specialmente se avete una configurazione hardware spinta, con un hard disk veloce. In tal caso provate a rinominare un file di una diecina di megabyte con l'estensione XML e copiarlo nella directory monitorata.

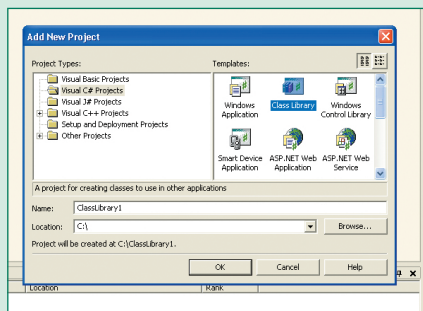
NOTA

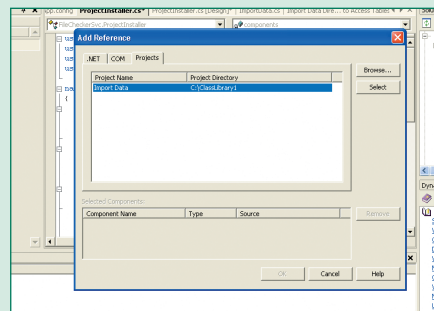
INSTALLARE L'ESEMPIO

Apriamo "Visual Studio.NET 2003 Command Prompt" dal menu start di Windows e lanciamo il comando InstallUtil.exe specificando il percorso e il nome dell'eseguibile FileCheckSvc.exe

UN SERVIZIO CHE IMPORTA IN UN DB I DATI SCRITTI IN UN FILE XML

CREIAMO IL PROGETTO

1 Da Visual Studio, usando File/New Project/ creaiamo un Windows Service, C# dal nome FileCheckerSVC. Rinominiamo in FileChecker.cs il file Service1.cs creato dal tool. Facendo doppio clic sul file Filecheckers.cs e utilizzando il tasto destro del mouse selezioniamo AddInstaller.

AGGIUNGIAMO UNA LIBRERIA

2 In questa libreria definiremo la classe ImportData che conterrà il metodo "Monitorizza", , la porzione di codice che controllerà se un file xml viene scritto sull'HD e lo importerà nel DB. Cliccare su File/Add Project/New Project /Class Library, salviamo la libreria come "Importdata". Rinominiamo in ImportData.cs il file class1.cs.

IMPORTIAMO LA LIBRERIA NEL PROGETTO

3 Per potere usare la libreria ImportData, il suo riferimento deve essere aggiunto al progetto. Per farlo usiamo il tasto destro nel solution explorer su "FileCheckerSvc" e scegliamo "Add Reference" infine nella tabsheet project scegliamo ImportData, e ovviamente terminiamo con OK.

messaggio all'interno del *Listener.* Questo semplice codice da origine ad un risultato rappresentato in **Figura 1** e **Figura 2**.Una riga viene aggiunta al log di eventi di sistema e viene specificato che *MiaApplicazione* è la fonte *(source)* che ha inviato i dati (**Figura 1**). Facendo doppio click sulla riga viene visualizzata una dialog box con ulteriori informazioni tra le quali la descrizione del messaggio inviato dall'applicazione.

TextWriterTraceListener

L'ultima classe messa a disposizione dal .NET Framework permette di specificare un file di testo oppure una console MS-DOS dove reindirizzare i messaggi tracciati dall'applicazione. Nel codice che segue è possibile osservare come implementare entrambe le scelte.

```
TextWriterTraceListener tw =
            new TextWriterTraceListener(Console.Out);
Trace.Listeners.Add(tw);
Trace.WriteLine("Entrato nel loop di MioMetodo()");
TextWriterTraceListener tw2 = new
TextWriterTraceListener(@"C:\MiaApplicazione.log");
Trace.Listeners.Add(tw2);
Trace.WriteLine("Entrato nel loop di MioMetodo()");
Trace.Flush();
Trace.Close();
```

Nel primo caso al costruttore della classe *TextWriterTraceListener* gli viene fornito l'oggetto *Console* che permette di stampare su una console DOS le informazioni, mentre nel secondo caso gli viene specificato il nome del file di log completo di percorso. È da notare che avendo aggiunto due nuovi Listener alla collezione entrambi saranno utilizzati durante il tracing.

CONCLUSIONI

Abbiamo visto come un programma teoricamente funzionante possa nascondere dei casi di malfunzionamento durante la sua esecuzione, praticamente impossibili da intercettare durante una fase di debugging.

Fabio Claudio Ferracchiati

AVVIARE E PROVARE L'ESEMPIO

Dal menu di Windows, Strumenti di amministrazione, eseguiamo sia l'event viewer che i servizi. Da quest'ultimo troviamo il servizio "File Checker Service" e avviamolo. Nel visualizzatore degli eventi, all'interno della sezione Applicazioni, verranno aggiunte delle righe che informeranno l'amministratore che il servizio è partito ed è in attesa di file XML.
Copiamo il file import.xml presente con il codice allegato alla rivista all'interno della directory monitorata. Facciamo un refresh dell'Event Viewer e scopriremo che il servizio è stato avviato correttamente e i dati importati nel database.
Attenzione però se copiate il file import_grande.xml all'interno dell'hard disk scoprirete che si verifica un errore e che questo errore viene scritto nell'event Viewer. Infatti se il file è molto grande, il processo di import dei dati inizia prima che il sistema operativo abbia finito di leggere l'intero file, scatenando un errore.

ATTIVIAMO IL LISTENER

```
public FileChecker()
{  InitializeComponent();
   el = new EventLogTraceListener(
                  @"File Checker Service");
   Trace.Listeners.Add(el);
   Trace.Listeners.RemoveAt(0);
   bs = new BooleanSwitch(
    "ImportDataSwitch", "Attiva/Disattiva il
                                   tracing");
}
```

4 **Nel costruttore della classe viene attivato il listener e l'output viene indirizzato sull'event viewer di Windows. Inoltre viene inizializzatto lo switch che ci servirà per attivare o disattivare il tracing a seconda delle modalità con cui distribuiremo il nostro programma.**

ATTIVARE E DISATTIVARE IL LISTNER

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.diagnostics>
    <switches>
      <add name="ImportDataSwitch"
                              value="1" />
    </switches>
  </system.diagnostics>
</configuration>
```

5 **Dal menu Project/Add New Item /Application Configuration aggiungiamo un file App.Config. Questo file sarà un file XML che consentirà a noi e eventualmente agli utilizzatori di settare un valore per lo Switch Bs che attiva o disattiva la modalità di Tracing, a seconda delle esigenze.**

ASSEMBLIAMO IL TUTTO

```
protected override void OnStart(
                            string[] args)
{
  Trace.WriteLineIf(bs.Enabled, DateTime
               .Now.ToShortDateString() +
                  ": Avviato il servizio");
  ImportData.ImportData objID = new
              ImportData.ImportData(bs);
  objID.Monitorizza();
}
```

6 **Questo codice va aggiunto a FileChecker.cs. All'avvio del servizio viene richiamato il metodo Monitorizza definito nella class ImportData, inoltre viene scritto un messaggio nell'event viewer che tiene traccia del corretto avvio del servizio. Date uno sguardo al box che contiene il codice del metodo "Monitorizza".**

Utilizzare l'Updater Application Block

Software sempre all'ultimo grido

In questo articolo aggiugeremo ad un'applicazione la capacità di autoaggiornarsi scaricando gli update direttamente dal Web con uno strumento gratuito di casa Microsoft

Utilizza questo spazio per le tue annotazioni

Vogliamo realizzare applicazioni che si autoaggiornano, un po' come fa Windows XP. Il nostro scopo è produrre il codice necessario affinché un'applicazione controlli se sono stati rilasciati eventuali aggiornamenti. Se sono stati rilasciati, l'applicazione deve scaricarli, installarli e aggiornare la propria versione a quella appena scaricata. Per realizzare tutto questo non è necessario scrivere tutto il codice a partire da zero, ma è possibile utilizzare uno scheletro preesistente prodotto da Microsoft e reso disponibile su GotDotNet all'indirizzo *http://www.gotdotnet.com/Community/Workspaces/workspace.aspx?id=83c68646-befb-4586-ba9f-fdf1301902f5*. Il file da considerare è *MicrosoftUpdaterApplicationBlock.zip*, al cui interno è contenuto un file *.msi* che consente l'installazione guidata del pacchetto sul proprio computer. Dopo avere installato l'Updater Application Block ci ritroveremo sul computer una cartella che contiene diverse soluzioni per la creazione di un'applicazione autoaggiornante. Quella che considereremo qui è una soluzione Visual Basic. Perciò, se avete scaricato e installato l'Updater, la prima cosa da fare è aprire dalla cartella VB la soluzione *Vb.Microsoft.Application. BlocksUpdater.sln*.

ARCHITETTURA DELL'UPDATER APPLICATION BLOCK

L'idea dell'Updater Application Block è che l'applicazione principale debba essere "lanciata" da un'altra applicazione: l'*AppStart*. *AppStart.exe* legge dal suo file di configurazione il path e la versione corrente dell'applicazione da lanciare, e la lancia, verificando che non sia già in esecuzione. Si occupa di effettuare gli aggiornamenti confrontando alcuni parametri contenuti nel suo file di configurazione. Quando viene scaricata la versione aggiornata, viene aggiornato anche il file di configurazione, in modo che *AppStart.exe* possa lanciare la versione giusta. Il file di configurazione ha la seguente struttura:

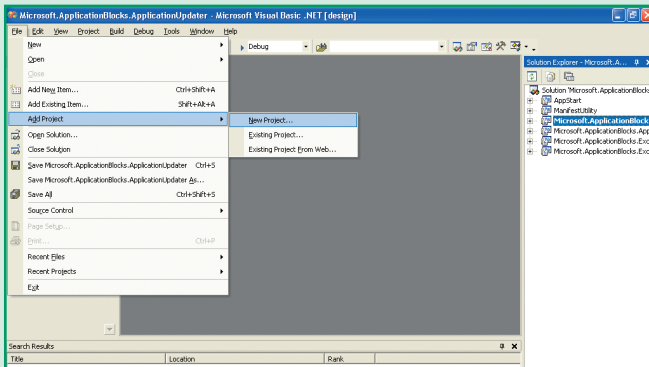
Fig. 1: Directory principale: l'applicazione di Start

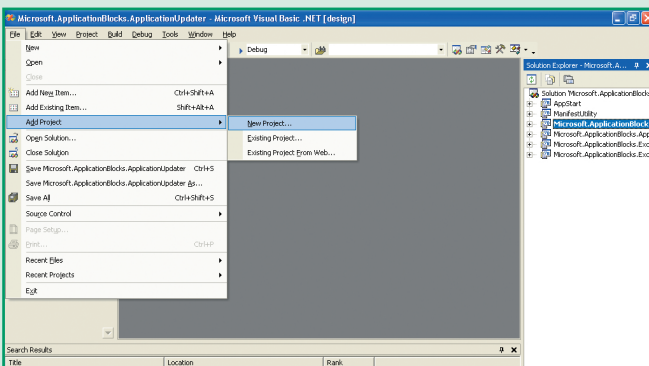
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="appStart" type=
      "Microsoft.ApplicationBlocks.ApplicationUpdater
        .AppStart.ConfigSectionHandler,AppStart" />
</configSections>
  <appStart>
    <ClientApplicationInfo>
    <appFolderName>C:\ioProgrammo_AutoUpdater
                   \1.0.0.0</appFolderName>
    <appExeName>FileSystemMonitor.exe
                               </appExeName>
    <installedVersion>1.0.0.0</installedVersion>
    <lastUpdated>2004-09-30T10:00:18.123448-04:00
                               </lastUpdated>
</ClientApplicationInfo>
  </appStart>
</configuration>
```

Viene definita la cartella che contiene l'applicazione da lanciare, il nome del file, la versione corrente e la data dell'ultimo aggiornamento.
Questi dati verranno aggiornati dall'applicazione in caso di disponibilità di nuove versioni. Va da sé che, per poter funzionare, le

Fig. 2: Directory con la versione iniziale del programma

IL CODICE PER GESTIRE L'AGGIORNAMENTO

AGGIUNGERE UN PROGETTO ALLO SCHELETRO

1 **Dopo avere caricato la soluzione VB *Microsoft.ApplicationBlocks.ApplicationUpdater*, aggiungiamo un nostro progetto cliccando su *File/Add Project/New Project*. Il nostro progetto sarà composto da una Form con un menu che contiene un item *"Check For Updates"***

AGGIUNGIAMO I RIFERIMENTI AL PROGETTO

2 **Clicchiamo con il tasto destro sul file del nostro progetto, poi su *"Add Refence"* e nella tabsheet projects della finestra di dialogo selezioniamo *Microsoft.Applications.ApplicationUpdater*. Una volta aggiunto il riferimento controlliamo nelle property che *"Copy Local"* sia settata a *true***

GESTIAMO L'AGGIORNAMENTO

```
Imports Microsoft.ApplicationBlocks. ApplicationUpdater
Imports System.Threading
Private Sub MenuItemCheckUpdates_Click(ByVal sender As
          System.Object, ByVal e As System.EventArgs) Handles
                                  MenuItemCheckUpdates.Click
  If Me.MenuItemCheckUpdates.Checked = False Then
    AddHandler AppDomain.CurrentDomain.ProcessExit, AddressOf
                                  CurrentDomain_ProcessExit
    AddHandler Me.Closed, AddressOf FormMain_Closed
    _updater = New ApplicationUpdateManager
    AddHandler _updater.DownloadStarted, AddressOf
                                  OnUpdaterDownloadStarted
    [ALTRI EVENTI]
    _updaterThread = New Thread(New ThreadStart(AddressOf
                                  _updater.StartUpdater))
    _updaterThread.Start()
    Me.MenuItemCheckUpdates.Checked = True
  Else
     StopUpdater()
     Me.MenuItemCheckUpdates.Checked = False
```

```
  End If
End Sub
```

3 **Vengono gestiti qui tutti gli Handler che spieghiamo nel dettaglio all'interno dell'articolo. Si noti che l'aggiornamento viene gestito in un Thread quindi viene considerata anche l'ipotesi che l'applicazione termini prima che l'aggiornamento venga completato.**

DEFINIAMO LE AZIONI

```
Private Sub OnUpdaterUpdateAvailable(ByVal sender As Object, ByVal
                                 e As UpdaterActionEventArgs)
Me.Invoke(New MarshalUpdEventToUIThread(AddressOf
 Me.OnUpdaterUpdateAvailableHandler), New Object() {sender, e})
End Sub
Private Sub OnUpdaterUpdateAvailableHandler(ByVal sender As
                      Object, ByVal e As UpdaterActionEventArgs)
Dim message As String
message = String.Format("Update available: The new version on
   the server is {0} and current version is {1} would you like to
         upgrade?", e.ServerInformation.AvailableVersion,
             ConfigurationSettings.AppSettings("version"))
If MessageBox.Show(message, "Update Available",
             MessageBoxButtons.YesNo) = DialogResult.No Then
   _updater.StopUpdater(e.ApplicationName)
   Me.StatusBar1.Text = "Update Cancelled"
Else
   Me.StatusBar1.Text += "Update in progress..."
End If
End Sub
```

4 **Per ogni handler è necessario definire una procedura di gestione. Qui mostriamo solo la procedura che si avvia se l'aggiornamento è disponibile. Potete visionare le altre nel corpo dell'articolo, oppure consultando il codice allegato. Si noti che l'avvio della procedura è affidato al Marshaling**

SISTEMIAMO LA CONFIGURAZIONE

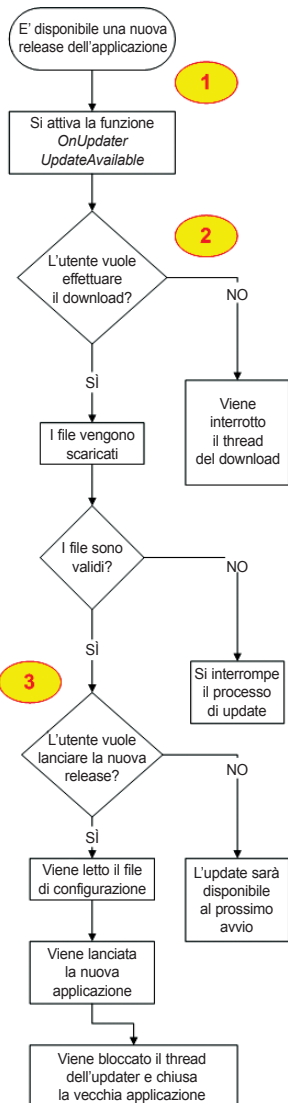
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="appStart"
      type="Microsoft.ApplicationBlocks.ApplicationUpdater
             .AppStart.ConfigSectionHandler,AppStart" />
  </configSections>
<appStart>
  <ClientApplicationInfo>
    <appFolderName>C:\ioProgrammo_AutoUpdater
                            \1.0.0.0</appFolderName>
    <appExeName>FileSystemMonitor.exe</appExeName>
    <installedVersion>1.0.0.0</installedVersion>
    <lastUpdated>2004-09-30T10:00:18.123448-04:
                                 00</lastUpdated>
      </ClientApplicationInfo>
    </appStart>
</configuration>
```

5 **Un file di configurazione minimo associato all'applicazione è quello di cui sopra. In realtà ci sono molte più configurazioni disponibili. Consultate il progetto allegato per visionare una versione più completa. Con questo *App.Config* definiamo la versione 1.0.0.0 della nostra applicazione.**

COMPILARE E DISTRIBUIRE

```
C:\ioProgrammo_AutoUpdater
  ├ AppStart.exe
  ├ AppStart.exe.config
  └ 1.0.0.0
      ├ FileSystemMonitor.exe
      ├ FileSystemMonitor.exe.config
      ├ Microsoft.ApplicationBlocks.ApplicationUpdater.dll
      ├ Microsoft.ApplicationBlocks.ApplicationUpdater.Interfaces.dll
      ├ Microsoft.ApplicationBlocks.ExceptionManagement.dll
      ├ Microsoft.ApplicationBlocks.ExceptionManagement.Interfaces.dll
      └ vbpowerpack.dll
```

6 Nel distribuire l'applicazione abbiate cura che la struttura su File System rispecchi quella che vi proponiamo. L'applicazione *AppStart.exe* sfrutta il file system per leggere i file di configurazione e capire se l'applicazione è da aggiornare o meno.

In figura potete seguire il diagramma di flusso dell'applicazione nelle operazioni di aggiornamento

applicazioni debbano essere distribuite rispettando una gerarchia particolare nella loro disposizione nel file system.

GESTIAMO GLI EVENTI DELL'AGGIORNAMENTO

La procedura di aggiornamento genera i seguenti eventi:

- **ServerManifestDownloaded** – indica che si è riusciti a scaricare il file manifest dal server
- **UpdateAvailable** – indica che è disponibile una nuova versione
- **DownloadStarted** – indica che è iniziato il download
- **DownloadCompleted** – indica che è finito il download
- **FileValidated** – indica che i file sono stati validati con successo
- **FileValidationFailed** – indica che i file non sono stati validati con successo

Questi eventi vengono generati da un thread diverso da quello principale dell'interfaccia utente, e quindi bisogna preparare due event handler: il primo gestisce l'evento, e lo gira al secondo che lo gestisce nel thread dell'interfaccia utente.
Si deve preparare quindi un *Delegate* per passare da un handler all'altro:

```
Private Delegate Sub MarshalUpdEventToUIThread(
        ByVal sender As Object, ByVal e As
        UpdaterActionEventArgs)
```

Se è presente una nuova versione, viene chiesto all'utente se la vuole scaricare o meno:

```
Private Sub OnUpdaterUpdateAvailable(ByVal sender As
        Object, ByVal e As UpdaterActionEventArgs)
    Me.Invoke(New MarshalUpdEventToUIThread(
        AddressOf Me.OnUpdaterUpdateAvailableHandler),
        New Object() {sender, e})           ' (1)
End Sub
Private Sub OnUpdaterUpdateAvailableHandler(
        ByVal sender As Object, ByVal e As
        UpdaterActionEventArgs)
Dim message As String
message = String.Format("Update available:  The new
    version on the server is {0} and current version is
    {1} would you like to upgrade?", e.ServerInformation
    .AvailableVersion, ConfigurationSettings.AppSettings(
    "version"))
If MessageBox.Show(message, "Update Available",
    MessageBoxButtons.YesNo) = DialogResult.No Then
    _updater.StopUpdater(e.ApplicationName)     ' (2)
    Me.StatusBar1.Text = "Update Cancelled"
Else
    Me.StatusBar1.Text += "Update in progress..."
End If
End Sub
```

Si vede che in caso l'utente scelga di non aggiornare, il thread che scarica gli aggiornamenti viene interrotto. Si potrebbe pensare che questo thread stia continuando a lavorare, ma viene tenuto bloccato dalla *Invoke* in attesa della risposta dell'utente, e dell'uscita dall'altro event handler (d'ora in poi non verranno più mostrati gli event handler di supporto per passare dal thread secondario al thread principale). Quando i file sono arrivati sul client e sono stati validati con successo, viene chiesto all'utente se vuole lanciare la nuova applicazione e chiudere la vecchia:

```
Private Sub OnUpdaterFilesValidatedHandler(ByVal
    sender As Object, ByVal e As UpdaterActionEventArgs)
Me.StatusBar1.Text = String.Format("FilesValidated
    successfully for application '{0}' ", e.ApplicationName)
If MessageBox.Show("Would you like to stop this
    application and open the new version?", "Open New
    Version?", MessageBoxButtons.YesNo) =
    DialogResult.Yes Then                      ' (3)
    StartNewVersion(e.ServerInformation)
End If
End Sub
Private Sub StartNewVersion(ByVal server As
    ServerApplicationInfo)
Dim doc As XmlDocument = New XmlDocument
Dim basePath As String
doc.Load(AppDomain.CurrentDomain.SetupInformation
    .ConfigurationFile)
basePath = doc.SelectSingleNode("configuration/
    appUpdater/UpdaterConfiguration/application/client
    /baseDir").InnerText
Dim newDir As String = Path.Combine(basePath,
    "AppStart.exe")
Dim newProcess As New ProcessStartInfo(newDir)
```

```
newProcess.WorkingDirectory = newDir
    + server.AvailableVersion Process.Start(newProcess)
CurrentDomain_ProcessExit(Nothing, Nothing)
Environment.Exit(0)
End Sub
```

Se l'utente sceglie di chiudere l'applicazione, viene letto il file di configurazione ed estratta la cartella base, che per il nostro setup DEVE contenere l'applicazione *AppStart.exe*, che verrà lanciata. A questo punto verrà bloccato il thread dell'updater e terminata l'applicazione. Gli altri eventi sono stati mappati su event handler che stampano nella status bar l'andamento dell'operazione di aggiornamento. Tutti questi event handler vanno naturalmente aggiunti nell'event handler del menu *"Check for updates…"*:

```
AddHandler _updater.DownloadStarted, AddressOf
                              OnUpdaterDownloadStarted
AddHandler _updater.FilesValidated, AddressOf
                              OnUpdaterFilesValidated
AddHandler _updater.UpdateAvailable, AddressOf
                              OnUpdaterUpdateAvailable
AddHandler _updater.DownloadCompleted, AddressOf
                              OnUpdaterDownloadCompleted
AddHandler _updater.FilesValidationFailed, AddressOf
                              OnUpdaterFileValidationFailed
AddHandler _updater.ServerManifestDownloaded,
  AddressOf OnUpdaterServerManifestDownloadedHandler
```

DETTAGLIO DELL'ESEMPIO

Al **PASSO 3** abbiamo definito gli handler che possono scatenarsi sulla base di un evento. E' importante gestire la situazione in cui l'applicazione viene terminata. Infatti, terminando l'applicazione, il thread che gestisce l'update deve anche esso essere terminato. Di fatto questa operazione viene gestita con

```
AddHandler AppDomain.CurrentDomain.ProcessExit,
                    AddressOf CurrentDomain_ProcessExit
AddHandler Me.Closed, AddressOf FormMain_Closed
```

Poi implementiamo gli event handler:

```
  Private Sub StopUpdater()
      _updater.StopUpdater()
      If Not (_updaterThread Is Nothing) Then
          Dim isThreadJoined As Boolean =
           _updaterThread.Join(UPDATERTHREAD_JOIN_
                                            TIMEOUT)
          'Se si riesce a fermarlo bene… se no lo si
                                          interrompe!
          If Not isThreadJoined Then
              _updaterThread.Interrupt()
          End If
          _updaterThread = Nothing
      End If
  End Sub
  Private Sub FormMain_Closed(ByVal sender As
                 Object, ByVal e As System.EventArgs)
      StopUpdater()
  End Sub
  Private Sub CurrentDomain_ProcessExit(ByVal
               sender As Object, ByVal e As EventArgs)
      StopUpdater()
  End Sub
```

CONFIGURAZIONE FINALE DEL CLIENT

Le modifiche al client sono finite, bisogna solo preparare il file di configurazione che verrà usato dall'Application Block per fare il suo lavoro. L'applicazione aveva già un suo file di configurazione che verrà esteso. Prima di tutto verrà aggiunta la versione corrente tra gli *appSettings*:

```
<add key="version" value="1.0.0.0" />
```

Poi verrà aggiunta una nuova sezione di configurazione contenente alcuni parametri di configurazione, come il path del file di log, l'intervallo di polling del thread per il controllo delle versioni, le informazioni sui downloader, sull'eventuale validator e sull'applicazione vera e propria da aggiornare:

```
<appUpdater>
  <UpdaterConfiguration>
   <polling type="Seconds" value="120" />
   <logListener logPath="C:\ioProgrammo_
                       AutoUpdater\UpdaterLog.txt" />
   <downloader .... /> <validator  .... /> <application
     name="FileSystemMonitor" useValidation="true">
<client>
<baseDir>C:\ioProgrammo_AutoUpdater</baseDir>
<xmlFile>C:\ioProgrammo_AutoUpdater\AppStart.exe.
config</xmlFile>
<tempDir>C:\ioProgrammo_AutoUpdater\newFiles</te
mpDir>
</client>
<server>
<xmlFile>
http://locahost/FileSystemMonitor_AutoUpdater/Server
Manifest.xml
</xmlFile>
<xmlFileDest>
C:\ioProgrammo_AutoUpdater\ServerManifest.xml
</xmlFileDest>
<maxWaitXmlFile>60000</maxWaitXmlFile>
```

DOVE SCARICARE L'UPDATER APPLICATION BLOCK

In questa pagina si possono trovare sia l'installer principale, sia la patch per il BITS 2.0
http://www.gotdotnet.com/workspaces/releases/viewuploads.aspx?id=83c68646-befb-4586-ba9f-fdf1301902f5

GLI APPLICATION BLOCK SONO GRATIS?

Si, sono completamente gratuiti, disponibili in forma sorgente e completamente modificabili e adattabili.

ESTENSIBILITÀ

Il design dell'Updater Application Block è completamente estensibile. Si possono creare nuovi downloader per nuovi canali, nuovi validator, ed è possibile aggiungere a mano le parti che mancano in maniera molto semplice. Sulla Message Board del workspace sono presenti vari post che parlano di estensioni già pronte.

INDIRIZZI UTILI

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/updater.asp

http://blogs.msdn.com/duncanma/archive/2004/01/14/58715.aspx

MIGLIORIE

Una prima miglioria è stata quella di settare la proprietà Checked del menu "Check for updates..." e lanciare l'update solo se vale False, altrimenti l'update viene fermato. Il tool può essere notevolmente migliorato:

- **Mancano gli eventi di download fallito e di nuova versione non disponibile, per permettere di gestire da codice lo stop del controllo in caso sia risultato negativo.**
- **I file di log rimangono all'infinito sul client, così come le cartelle con le versioni precedenti.**
- **Se si crea un setup con Windows Installer, bisogna impostare di non ripristinare la situazione iniziale nel caso si cancelli la cartella con la versione vecchia del programma. Tutti i path sono assoluti, la cosa migliore è creare dei passi che "aggiornano" i file di configurazione in runtime, settando i path giusti.**

Lorenzo Barbieri è laureato in Ingegneria Informatica e lavora come Trainer e Consulente sulle tecnologie Microsoft. È Microsoft Certified Trainer ed è certificato MCSD.NET e MCDBA su SQL Server 2000.

```
</server>
</application>
</UpdaterConfiguration>
</appUpdater>
```

Bisogna specificare i path della cartella base del programma, del file di configurazione e di una cartella temporanea che verrà usata per contenere i file scaricati. La configurazione della comunicazione col server richiede il path del manifest sul server, il path dove verrà salvato sul client e il timeout.

CONFIGURAZIONE DEL SERVER

Per configurare il server bisogna creare una virtual directory con lo stesso nome usato nel file config del client (*FileSystemMonitor_AutoUpdater* nel nostro caso) e impostare l'accesso anonimo, e le seguenti impostazioni devono essere selezionate:

- Script source access
- Read
- Write
- Directory Browsing

Per permettere il download degli *App.config* da Web, bisogna andare in Configuration, selezionare .config e rimuoverlo dai mapping predefiniti.
In questa virtual directory dobbiamo creare una cartella con il nome della nuova versione, supponiamo la 2.0.0.0. Nella cartella dobbiamo mettere tutti i file necessari al funzionamento della nuova versione, quindi anche tutte le DLL e i file di configurazione. È necessario a questo punto creare il file manifest. Per farlo, possiamo selezionare in Visual Studio .NET la *ManifestUtility*, impostarla come progetto di default e lanciarla. Come directory selezioniamo la directory chiamata *2.0.0.0* che abbiamo creato prima, poi impostiamo il path che useremmo via Web, e la versione corrispondente dell'applicazione. Se vogliamo far validare i file scaricati facciamo generare una coppia di chiavi

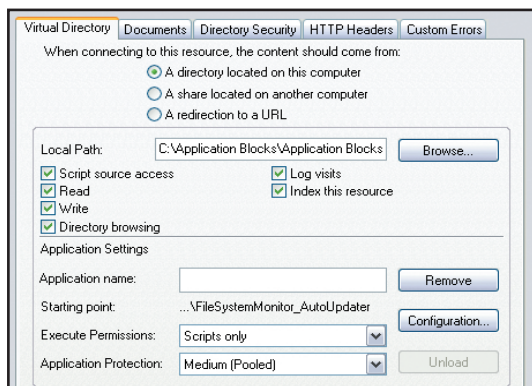



Fig. 3: **Proprietà virtual directory**

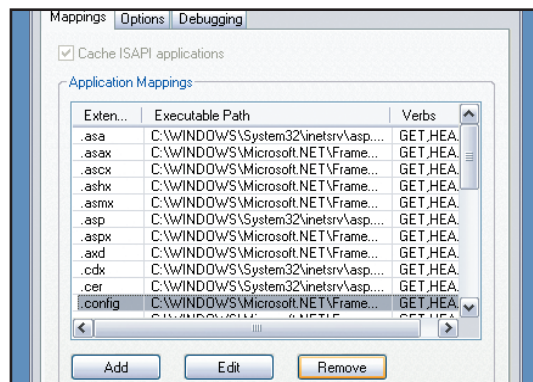



Fig. 4: **Rimozione di .config**

pubblica/privata dal menu del programma, le salviamo e poi importiamo la chiave privata con l'apposito bottone. La chiave pubblica va copiata nel file di configurazione del programma. Come Validator Assembly selezioniamo *ApplicationUpdater.dll* e come Class selezioniamo *RSAValidator*. Non avendo bisogno di un post-processore, lo deselezioniamo, e poi possiamo salvare il manifest nella root della nostra virtual directory.

TEST DELL'AGGIORNAMENTO

Una volta configurato il sistema, si può lanciare *AppStart.exe* per verificare l'autoaggiornamento della propria applicazione. Ci sono alcune considerazioni nel caso l'aggiornamento non vada a buon fine:

- **URLScan** – nella configurazione di default URLScan blocca le URL con dentro il "." (punto), e blocca anche i file ".exe". Bisogna quindi intervenire su *URLScan* per rilasciare queste impostazioni, oppure disabilitare *URLScan* per solo per la virtual directory.

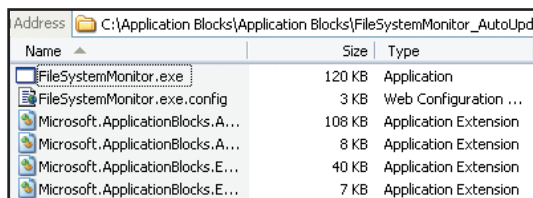



Fig. 5: **File nella cartella 2.0.0.0**

- **BITS 2.0** – se avete installato l'update per BITS 2.0, l'Application Block non funziona più. Bisogna scaricare la fix dal workspace GDN, fix che per ora è disponibile solo per la versione C#. Nessuno ci vieta di utilizzare la versione C# del sorgente, al posto della versione VB.NET inclusa nell'esempio

Lorenzo Barbieri

Visualizzare il tempo trascorso davanti al computer

Grafici con MSChart

Vedremo come utilizzare i controlli MSChart e MSHFlexGrid per visualizzare i dati di un foglio Excel e creeremo un grafico dei tempi di esecuzione dei processi Windows

La gestione dei processi può essere fatta con varie tecniche, in quest'articolo si è preferito utilizzare quella basata sulle funzioni *API CreateProcess* e *TerminateProcess*. Delle applicazioni sono presentate solo le routine più importanti, perciò è importante leggere l'articolo avendo di fronte il codice allegato sul CD.

MSCHART ED EXCEL

Iniziamo descrivendo come caricare i dati del grafico da un file Excel. Il file nominato *spese.xls* contiene i dati del bilancio familiare suddivisi per mese e categoria; per semplificare, però, i dati sono anche raggruppati per quadrimestre. Le procedure che si occupano di caricare i dati dal file sono la *mnuexcel* e la *apriexcel*, la prima è relativa alla voce di menu *carica dati* da Excel.

```
Private Sub mnuexcel_Click()
 apriexcel
End Sub
Private Sub apriexcel()
Dim AppExcel As Object
Dim FoglioExcel As Object
On Error Resume Next
Set AppExcel = GetObject(, "Excel.Application")
If Err.Number <> 0 Then
 Set AppExcel = CreateObject("Excel.Application")
End If
Set FoglioExcel = AppExcel.Workbooks.Open(App.Path
                                    & "\spese.xls")
Dim colonnaEx As Integer
Dim rigaEx As Integer
Dim letterecolonneEx
letterecolonneEx = Array("B", "C", "D")
For colonnaEx = 1 To 3
 For rigaEx = 18 To 21
  ArrDati(colonnaEx, rigaEx - 17) =
                    FoglioExcel.Worksheets(1). _
  Range(letterecolonneEx(colonnaEx) + CStr(rigaEx))
 Next
Next
` inserire istruzioni della MnuQua_Click
MSChart1.Title = "Spese per quadrimestre Excel"
End Sub
```

Fig.1: ***Il form FrmMSChart e il foglio Excel Spese.xsl***

Nella apriexcel dopo aver avviato Excel, con la *GetObject* o la *CreateObject*, sono lette i valori delle celle (del file *spese.xsl* comprese tra le colonne *B-D* e le righe 18-21) ed inserite nella matrice *ArrDati*.

MSCHART E MSHFLEXGRID

In questo paragrafo si descrive come passare i dati della griglia del controllo *MSChart1* ad un controllo *MSHFlexGrid*. Per implementare questo esempio, nel progetto, è stato inserito un form, nominato *FrmFlex*, con all'interno una *MSHFlexGrid*. In pratica partendo dai valori contenuti in un grafico tente-

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
caratteristiche di base dei controlli MSHFlexGrid e MSChart, degli oggetti ADO e delle API.

Software
Windows 98 o superiore - Visual Basic 6 SP6.

Impegno

Tempo di realizzazione

remo di riempire una griglia

```
Private Sub MnuGrigliaDati_Click()
 FrmFlex.Show
End Sub
```

Le istruzioni per impostare e caricare la *MSHFlex-Grid* sono inserite nella *Form_Load*.

```
Private Sub Form_Load()
 Dim chart As MSChart
 Dim colonna As Integer
 Dim comcol, comrig As Integer
Const ampiezzacolonna = 1500
Const ampiezzariga = 260
Dim riga As Integer
Set chart = FrmMschart.MSChart1
'1) imposta dimensione griglia
MSHFlexGrid1.Cols = chart.DataGrid.rowCount + 1
MSHFlexGrid1.Rows = chart.DataGrid.columnCount + 1
MSHFlexGrid1.ColWidth(0) = ampiezzacolonna
MSHFlexGrid1.RowHeight(0) = ampiezzariga
'2) imposta etichetta righe
For colonna = 1 To chart.DataGrid.rowCount
 MSHFlexGrid1.TextMatrix(0, colonna) = _
 chart.DataGrid.RowLabel(colonna, 1)
```

NOTA

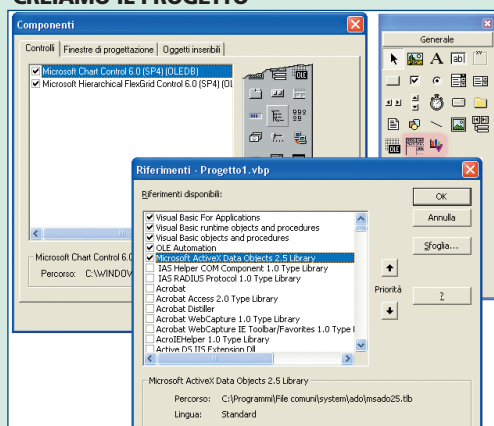
STATO DEL PROCESSO

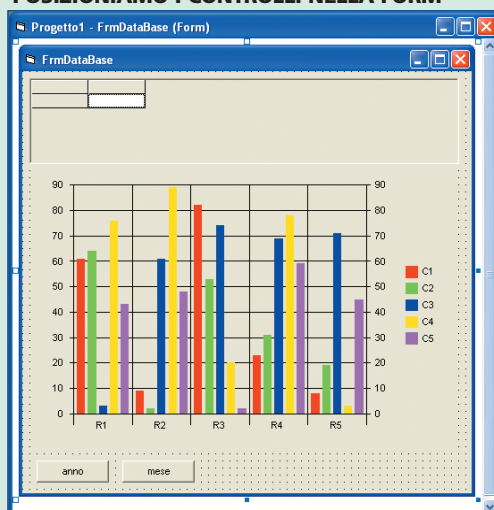
Per stabilire se un processo è terminato si possono utilizzare le funzioni API che permettono di recuperare il suo stato, cioè WaitForSingleObject e GetExitCodeProcess. E' utile valutare lo stato di un processo dato che può capitare che esso venga avviato con il form di controllo, ma terminato attraverso i pulsanti della finestra che lo contiene (e quindi i tempi dell'applicazione non sono più valutabili!).

FLEXGRID

Per poter utilizzare un controllo MSHFlexGrid è necessario referenziare il file MSHFlxGd.ocx. Il controllo Microsoft Hierarchical FlexGrid consente di visualizzare ed eseguire operazioni con i dati di tabelle anche correlate in gerarchie. Nelle celle di un controllo MSHFlexGrid è possibile inserire testo ed immagini. La MSHFlexGrid, inoltre, offre il massimo grado di flessibilità con le operazioni di ordinamento, unione e formattazione.

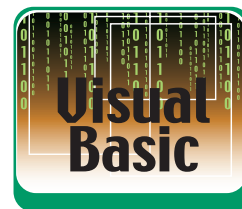
CREARE UN GRAFICO VB PARTENDO DA ACCESS

CREIAMO IL PROGETTO

1 **Creare un nuovo progetto e specificare i riferimenti alle librerie: MSCHRT20, MSHFlxGd e MSADO.**

POSIZIONIAMO I CONTROLLI NELLA FORM

2 **Sul Form1 disporre: un controllo MSChart, una MSHFlexGrid e due CommandButton. I due pulsanti, nominati mese e anno, servono ad invocare le routine che creano i grafici e caricano i dati nella MSHFlexGrid. In particolare, il pulsante mese crea un grafico con le spese dei mesi di un certo anno; anno, invece, valuta le spese complessive di ogni anno.**

GESTIAMO I PULSANTI

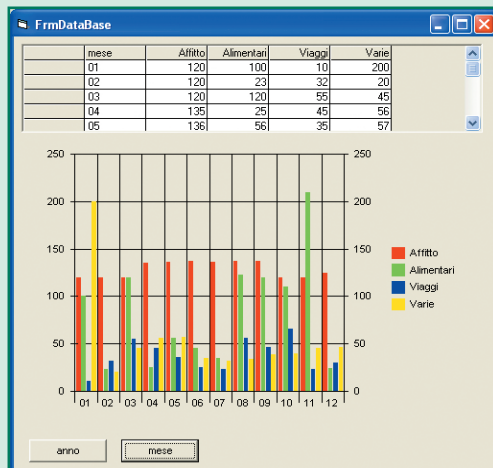
```
Dim RstSpese As ADODB.Recordset
Private Sub anno_Click()
`raggruppa per anno
 caricadati ("SELECT  Anno, sum(Affitto) AS "
   & " Affitto, sum(alimentari) AS Alimentari, sum(
         viaggi)" & " AS Viaggi, sum(varie) AS Varie
                 FROM spese GROUP BY anno;")
End Sub
Private Sub mese_Click()
`raggruppa per mese nel 2004
 caricadati ("SELECT Mese, sum(Affitto) AS "
   & " Affitto, sum(alimentari) AS Alimentari, sum(
        viaggi) AS " & " Viaggi, sum(varie) AS Varie
        FROM spese " & " WHERE anno=""2004""
                             GROUP BY mese;")
End Sub
Public Sub caricadati(query As String)
 Dim strcnn As String
 strcnn = "Provider=microsoft.jet.oledb.4.0;"
   & "Data Source=" + App.Path + "\spese.mdb; "
 `il database deve essere nella directory del progetto
 Set RstSpese = New ADODB.Recordset
 RstSpese.CursorType = adOpenKeyset
 RstSpese.LockType = adLockOptimistic
 RstSpese.Open query, strcnn, , , adCmdText
 RstSpese.MoveFirst
 Set MSChart1.DataSource = RstSpese
 Set MSHFlexGrid1.DataSource = RstSpese
End Sub
```

3 **La procedura di gestione dell'applicazione è quasi interamente contenuta in questo codice che gestisce gli eventi dei due pulsanti *anno* e *mese*. Inserire nel form le dichiarazioni e il codice illustrato di seguito cioè *anno_Click, mese_Click* e *caricadati.* Nelle procedure dei pulsanti è invocata la caricadati alla quale è passata una query SQL di aggregazione. Per semplificare, nella select che raggruppa i mesi, il campo anno è stato impostato sul 2004. Nella procedura *caricadati* per eseguire la query è usato il metodo *Open* del *RecordSet,* mentre per impostare i controlli è utilizzata la proprietà DataSource. Notare che la proprietà DataSource del controllo *MSChart* accetta solo rowset aggregati con *count(*), sum(*), min(*)* o *max(*).***

```
 MSHFlexGrid1.ColWidth(colonna) = ampiezzacolonna
Next colonna
'3) imposta etichetta colonne
For riga = 1 To chart.DataGrid.columnCount
 MSHFlexGrid1.TextMatrix(riga, 0) = _
 chart.DataGrid.ColumnLabel(riga, 1)
MSHFlexGrid1.RowHeight(riga) = ampiezzariga
Next riga
'4) imposta valori grafico
For colonna = 2 To MSHFlexGrid1.Cols
 For riga = 2 To Me.MSHFlexGrid1.Rows
  Me.MSHFlexGrid1.TextMatrix(riga - 1, _
  colonna - 1) = chart.ChartData(colonna - 1, riga - 1)
 Next
Next
'5) ridimensiona FlexGrid e form
MSHFlexGrid1.Move 0, 0
MSHFlexGrid1.Width = ampiezzacolonna *
                              MSHFlexGrid1.Cols +50
MSHFlexGrid1.Height = ampiezzariga *
                                 MSHFlexGrid1.Rows
Me.Width = MSHFlexGrid1.Width + (
                              Me.Width - Me.ScaleWidth)
Me.Height = MSHFlexGrid1.Height + (
                            Me.Height - Me.ScaleHeight)
End Sub
```

IL LATO ACCESS

4 **Creare e popolare, usando Access o Visual Data Manager, il database *Spese*. Come accennato in esso bisogna includere la tabella *Spese*, i cui campi sono descritti nella Tabella 1, notare che *Anno* e *Mese* costituiscono il campo *chiave*.**

Nome Campo	Tipo dato
Anno	Testo - 4 caratteri
Mese	Testo - 2 caratteri
Affitto	Valuta
Alimentari	Valuta
Viaggi	Valuta
Varie	Valuta

ESEGUIAMO L'APPLICAZIONE

5 **Eseguire il programma e cliccare i pulsanti anno e mese.**

Dopo le dichiarazioni viene copiato nella variabile *"chart"* l'oggetto *MSChart1* con la *FrmMschart.MSChart1*.
Poi s'impostano il numero di righe e colonne della *MSHFlexGrid* e le loro dimensioni; dato che in essi bisogna includere la riga e la colonna delle etichette questi valori sono incrementati di 1 rispetto a quelle dello *MSChart1*. Le dimensioni delle righe e delle colonne della griglia, e quindi della *MSHFlexGrid*, sono impostate sulla base delle costanti ampiezzacolonna e ampiezzariga. Con due cicli *For* separati, invece, sono impostate le etichette di riga e di colonna, mentre con due cicli di *For* innestati vengono impostati i valori dei punti del grafico. Infine, sulla base delle impostazioni precedenti, sono ridimensionati la *MSHFlexGrid* e il form.

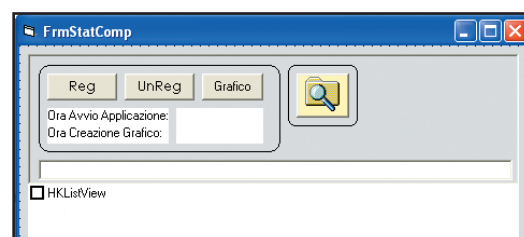



Fig. 2: *Il form di monitoraggio in fase di progettazione*

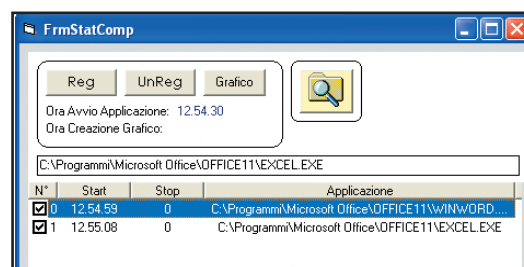



Fig. 3: *Il form con due applicazioni registrate*

IL GRAFICO DEI PROCESSI

Useremo ora una tecnica molto semplice per monitorare il tempo di utilizzo delle applicazioni, e di conseguenza il tempo trascorso davanti al computer. Si scelgono le applicazioni e si registra il tempo di avvio e di chiusura. Questo viene fatto con un form che contiene un ListView, 4 pulsanti, un textbox, delle Label, un CommonDialog e un PictureBox contenitore. Si controlli la **Figura 2** per capire come sono disposti gli elementi sul form. Il *ListView* è utilizzato per registrare i seguenti dati dell'applicazione: *path+nome*, tempo di avvio e tempo di chiusura. Un'applicazione da monitorare si sele-

ziona utilizzando il pulsante *Ricerca* (sul form il pulsante con icona). L'applicazione selezionata si registra nel *ListView* con il pulsante *Reg*. Dopo che l'applicazione è registrata può essere avviata o chiusa usando il *CheckBox* associato alla riga del *ListView*; queste operazioni tra l'altro registrano i tempi. Per cancellare un'applicazione dal *ListView* bisogna selezionare l'Item e cliccare il pulsante *UnReg*. Per creare il grafico, che mostra i tempi di utilizzo delle applicazioni, bisogna cliccare il pulsante *Grafico*. Inoltre all'avvio del form su una label (nominata labeltime) è registrato il tempo di avvio del monitoraggio, mentre quando è cliccato il pulsante *Grafico* su un'altra label (nominata labeltimegrafico) è registrato il momento in cui è stato creato il grafico.
Per gestire l'avvio e la chiusura, di un'applicazione, cioè di un processo, sono utilizzati diversi elementi dell'API di Windows, sommariamente, descritti nel paragrafo successivo.

COMANDO SHELL

Un modo alternativo, ma meno efficiente, per controllare i processi è basato sul commando Shell. Questo, crea un nuovo processo e restituisce il suo identificatore. La sintassi del comando è la seguente:

```
Shell(pathname[
,windowstyle])
```

Dove Pathname è il nome del programma da eseguire e Windowstyle è lo stile della finestra che mostra il programma.

PROCESSI, THREAD E APPLICAZIONI

Un processo, in parole povere, è un programma in esecuzione. In generale nel contesto di un processo sono in esecuzione uno o più thread, quest'ultimo è definito come l'unità con cui il sistema operativo alloca tempo di esecuzione. Per la creazione di un processo bisogna utilizzare la seguente funzione:

```
Public Declare Function CreateProcess Lib "kernel32" _
  Alias "CreateProcessA" (ByVal lpApplicationName As
      String, ByVal lpCommandLine As String, , , , , ,
  lpStartupInfo As STARTUPINFO, lpProcessInformation
              As PROCESS_INFORMATION) As Long
```

La *CreateProcess* crea un nuovo processo e il suo thread primario. Il nuovo processo esegue un'applicazione in un contesto di sicurezza. I parametri della funzione, utilizzati negli esempi sono: *lpCommandLine* che specifica l'applicazione da eseguire; *lpStartupInfo* un puntatore ad una struttura *Startupinfo; lpProcessInformation* un puntatore ad una struttura *ProcessInformation*. Basta sapere che la struttura *Startupinfo* specifica alcune informazioni sulla Window che mostra l'applicazione e che la *ProcessInformation* all'atto della creazione riceve informazioni sul nuovo processo e sul suo thread. I restanti parametri sono impostati utilizzando i seguenti elementi:

```
Public Const NORMAL_PRIORITY_CLASS = &H20&
Public sNull As String
```

La funzione che consente di terminare un processo e tutti i suoi thread è la seguente:

```
Public Declare Function TerminateProcess Lib
  "kernel32" (ByVal hProcess As Long, ByVal uExitCode
                                  As Long) As Long
```

Il parametro *hProcess* è l'identificatore del processo da terminare, questo dato è contenuto nella struttura *ProcessInformation* inizializzata con la *CreateProcess*. Con *uExitCode* invece si specifica il codice di uscita usato dai processi e dai thread come risultato di questa operazione. Notare che la fine di un processo non causa la fine di tutti i suoi processi figli e che la dichiarazione dei tipi deve precedere quella delle funzioni. Il dettaglio sui type, indicati sotto, si ricava con il Visualizzatore API per Visual Basic 6.

```
PROCESS_INFORMATION
 `dettagli nel CD
End Type
Public Type STARTUPINFO
 `dettagli nel CD
End Type
```

Dato che per ogni applicazione registrata sul *ListView* è necessaria una coppia *PROCESS_INFORMATION*, *STARTUPINFO*, nell'esempio si definiscono i seguenti array.

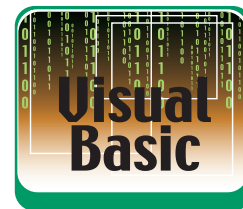
```
Option Base 0
`limite inferiore dell'array è Zero
Private Const NumeroApp = 100
`stabilisce il numero di applicazioni controllabili
Public ArraypSInfo(NumeroApp) As STARTUPINFO
Public ArraypPInfo(NumeroApp) As PROCESS_
                                   INFORMATION
```

IL FORM DI CONTROLLO

Nella parte dichiarativa deve essere definito l'indice per numerare le righe del *ListView*.

```
Dim numkey As Integer
```

Nella *Form_Load* vanno predisposte le istruzioni per impostare le caratteristiche del *ListView* e di un *PictureBox (Picture1)* che contiene gli altri elementi del form. La *Picture1* è inserita per rendere più semplice il ridimensionamento del form.

```
Private Sub Form_Load()
 Picture1.Appearance = 0
 Picture1.BorderStyle = 0
 TxtApplicazione.Appearance = 0
 Labeltime.Caption = Time
 … nel CD
 HKListView.Checkboxes = True
```

```
HKListView.View = lvwReport
HKListView.FullRowSelect = True
End Sub
```

Notare che nella *Form_Load* con *Labeltime.Caption = Time* s'imposta il tempo di avvio del form. Invece, le istruzioni per registrare un'applicazione nel *ListView* sono inserite nella *CmdReg* (… nel CD), in essa si incrementa il valore di numkey e si invoca la *AggiungiHK*.

```
Sub AggiungiHK(KeyAtom As Integer, hk As String, appli
                                                As String)
Dim itmX As ListItem
Set itmX = HKListView.ListItems. Add(, ,
CStr(KeyAtom))
    itmX.Checked = False
    itmX.SubItems(1) = 0
    itmX.SubItems(2) = 0
    itmX.SubItems(3) = TxtApplicazione
End Sub
```

La *AggiungiHK* crea una riga del *ListView* con i dati dell'applicazione selezionata. Notare che *itmX.SubItems (1)* e *itmX.SubItems(2)* sono i tempi di avvio e chiusura del processo e che *KeyAtom* è il valore che lo identifica, questo verrà utilizzato come indice degli Array dei type. La parte più importante del codice di controllo è quella correlata all'evento *ItemCheck* del *ListView,* dato che abilita i processi e i contatori del tempo di utilizzo.

```
Private Sub HKListView_ItemCheck(ByVal Item As
                                     MSComctlLib.ListItem)
 On Error GoTo errore
 If Item.Checked Then
  attivaAPP Item.ListSubItems.Item(3), Item.Text
  Item.ListSubItems.Item(1).Text = Time
  Item.ListSubItems.Item(2).Text = 0
 Else
  disattivaAPP Item.Text
  Item.ListSubItems.Item(2).Text = Time
 End If
 Exit Sub
errore:
Err.Clear
End Sub
```

Nella *HKListView_ItemCheck* in base al valore del *CheckBox*, dell'Item selezionato, viene invocata la *attivaAPP* o la *disattivaAPP,* presentate sotto.

```
Private Sub attivaAPP(nomeapp As String, keylist As Integer)
 sInfo.cb = Len(ArraypSInfo(keylist))
 ValoreRest = CreateProcess(sNull, nomeapp, ByVal
   0&, ByVal 0&, 1&, NORMAL_PRIORITY_CLASS, ByVal
   0&, sNull, ArraypSInfo(keylist), ArraypPInfo(keylist))
End Sub
```

La *AttivaAPP* in base al nome dell'applicazione e alla riga del *ListView* abilita il processo associato all'applicazione.

```
Sub disattivaAPP(keylist As Integer)
 ValoreRest = TerminateProcess(
                          ArraypPInfo(keylist).hProcess, 0&)
 ValoreRest =
CloseHandle(ArraypPInfo(keylist).hThread)
 ValoreRest =
CloseHandle(ArraypPInfo(keylist).hProcess)
End Sub
```

La disattivaAPP chiude il processo identificato dal valore di keylist. Nella *disattivaAPP* viene utilizzata, anche, la funzione API *CloseHandle*. La *CloseHandle* chiude l'oggetto il cui identificatore è passato come parametro e decrementa il contatore di handle utilizzato dal sistema operativo. Notare, dopo che l'ultimo handle di un oggetto è chiuso questo è rimosso dal sistema.

GRAFICO DEI TEMPI DI UTILIZZO

L'ultimo parte da presentare è quella che riguarda la creazione del grafico dei tempi di utilizzo delle applicazioni registrate nel *ListView.* A tal fine sul form è predisposto il pulsante *Grafico* con il seguente codice.

```
Private Sub Grafico_Click()
 Dim cont As Double
 Dim com As Date
 Dim data1 As Date
 Dim data2 As Date
 Labeltimegrafico.Caption = Time
 data1 = Labeltime.Caption
 data2 = Labeltimegrafico.Caption
 com = data2 - data1
 cont = CInt((Hour(com) * 60) + Minute(com) +
                                    (Second(com) / 60))
 'cont è espresso in minuti
 FrmMschart.caricadalistview HKListView, cont
End Sub
```

La *Grafico_Click* valuta il tempo trascorso dall'avvio del form e invoca la caricadalistview, procedura presente su *FrmMschart,* passandogli come parametri il *ListView* e il tempo trascorso dall'avvio del form. La *caricadalistview* costruisce un grafico sulla base dei tempi registrati nelle righe del *ListView.* Nel grafico creato, come etichetta delle colonne sono usati i nomi dell'applicazione mentre per l'etichetta di riga è usato il tempo totale trascorso dall'avvio del form.

Massimo Autiero

INTERAZIONE FORM EVENTI

In generale l'interazione tra form è implementata utilizzando gli eventi e distinguendo tra form sorgente e form destinatario. Nel form sorgente gli eventi sono dichiarati attraverso l'istruzione *public Event* seguita dal nome e dai parametri dell'evento. Nel form destinatario, invece, si utilizza la parola chiave *WithEvents* seguita dal nome di una variabile e da quello del form sorgente. Il form sorgente invia un evento utilizzando l'istruzione *RaiseEvent* seguita dal nome dell'evento e dai valori dei relativi parametri. Il form destinatario riceve l'evento, grazie a *WithEvents*, ed esegue del codice di risposta.

Utilizziamo JAI per costruire un programma Java di fotoritocco

Immagini con effetti speciali III parte

In questo articolo ottimizzeremo l'apertura dei file di immagine, ne implementeremo il salvataggio e visualizzeremo alcune importanti informazioni

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
Linguaggio Java

Software
Java2 SDK 1.4.2

Impegno

Tempo di realizzazione

Nei numeri precedenti abbiamo affrontato lo sviluppo di JAIPhoto, una semplice applicazione di manipolazione di immagini. Allo stato attuale, sono presenti le funzionalità di base. Queste includono l'apertura di file di immagine e l'applicazione di alcuni filtri. La finestra principale dell'applicazione mostra l'immagine che è stata aperta, offrendo la possibilità di selezionare il livello di zoom. Questo è pilotato da una combo box presente in basso nella finestra.

Fig. 1: La finestra principale dell'applicazione JAIPhoto

MIGLIORIAMO LE FUNZIONI DI APERTURA

Tramite la funzione di menu *File | Apri* è possibile aprire un file per la manipolazione. Ad oggi, però, il programma consente di aprire un qualsiasi file, anche di formato non supportato. I programmi più evoluti hanno invece la possibilità di limitare la selezione dei file ai formati riconosciuti, individuati dall'estensione. Anche in Java è possibile ottenere questo, specificando un oggetto *FileFilter* sul *JFileChooser* utilizzato per scegliere il file. *FileFilter* è una classe astratta di cui non è presente implementazione di default, che specifica due metodi:

- **boolean accept(File).** Ritorna *true* se il file passato come parametro rientra nei parametri di questo filtro;
- **String getDescription().** Ritorna una descrizione testuale utilizzata per mettere a conoscenza l'utente della tipologia di file filtrata da questo oggetto.

La libreria *JAI (Java Advanced Imaging)* supporta un insieme di formati di file, riassunti in **Tabella 1**. Verrà ora sviluppato un filtro in grado di rendere selezionabili solo file il cui tipo è tra quelli

formato	estensione
Immagine GIF	*.gif*
Immagine JPEG	*.jpeg, .jpg*
Immagine scanner (TIFF)	*.tiff, .tif*
Portable Network Graphics	*.png*
Windows bitmap	*.bmp*

Tabella 1: Formati di file supportati da JAI

presenti in tabella, verificando l'estensione del file. Per fare questo viene creata una sottoclasse anonima di *FileFilter* e passata all'oggetto *JFileChooser*. Il codice del metodo *apri()* diventa:

```
JFileChooser fc = new JFileChooser();
fc.setFileFilter( new FileFilter() {
  public boolean accept(File f) {
    return f.getName().endsWith(".gif") ||
    f.getName().endsWith(".jpg") ||
    f.getName().endsWith(".jpeg") ||
    f.getName().endsWith(".tif") ||
    f.getName().endsWith(".tiff") ||
    f.getName().endsWith(".png") ||
    f.getName().endsWith(".bmp");}
```

```
    public String getDescription() {
        return "Immagini supportate (*.gif, *,jpg, *.tiff,
                                         *.png, *.bmp)";
    }
} );
```

l'implementazione di *FileFilter* indica all'oggetto *JFileChooser* se il file è selezionabile ritornando *true* dal metodo *accept()*. Questo viene chiamato per ogni file presente nella directory in fase di visualizzazione, che viene passato come parametro sotto forma di oggetto *File*. Come si nota, l'implementazione del metodo sopra mostrata controlla l'estensione del file e ritorna true se questa equivale ad una delle estensioni supportate. Inoltre, la descrizione di questo filtro ritornata tramite il metodo *getDescription()*, deve essere indicativa del criterio di accettazione implementato.
In questo caso è stata adottata l'accortezza di enumerare le estensioni previste all'interno della descrizione, sorvolando sulle varianti di uno stesso tipo, come *.jpg* o *.jpeg*. Questo approccio ha però un problema. Nel caso infatti di nomi di file in maiuscolo, il confronto tra stringhe fallisce, ed il relativo file non risulta selezionabile. Per ovviare a questo problema è possibile convertire in tutte lettere minuscole il nome del file, per poi eseguire il confronto come di consueto. Il codice diventa dunque:

```
public boolean accept(File f) {
  String fileName = f.getName().toLowerCase();
  return fileName.endsWith(".gif") ||
  fileName.endsWith(".jpg") ||
  fileName.endsWith(".jpeg") ||
  fileName.endsWith(".tif") ||
  fileName.endsWith(".tiff") ||
  fileName.endsWith(".png") ||
  fileName.endsWith(".bmp");}
```

la descrizione ritornata da *getDescription()* è riportata nella combo box in fondo alla finestra di selezione file. Tramite questa combo è possibile scegliere tra i vari criteri disponibili. Ad esempio, JAI-Photo invece che una unica categoria di immagini avrebbe potuto avere:

- **Immagini Web** *(*.gif, *.jpg, *.png);*
- **Immagini Windows** *(*.bmp);*
- **Immagini Scanner** *(*.tiff).*

In alcuni casi, infatti decidere di suddividere i diversi file supportati in categorie diverse. Ciascuna delle quali è implementata da un oggetto FileFilter diverso. Per fare in modo che un unico JFileChooser possa accettare più FileFilter è necessario impostare questi oggetti utilizzando il metodo *addChoosableFileFilter()* invece che *setFileFilter()*.

UN PANNELLO INFORMATIVO

È utile aggiungere un ulteriore pannello che contenga informazioni specifiche sull'immagine, quali dimensioni, colori, ecc. Per prima cosa è utile definire il metodo che ritornerà l'intero pannello di visualizzazione, con caricate le informazioni da visualizzare. Il metodo si chiama *createInfoPanel()* e ritorna un *JScrollPane*. Si è scelto di ritornare un pannello con scroll invece che un normale pannello JPanel in quanto si vuole fornire di scorrimento il pannello, che potrebbe essere più grande dello spazio video a disposizione. All'inizio del metodo vengono estratte le informazioni sui sample e sui colori, per utilizzarle in seguito:

```
JScrollPane createInfoPanel() {
  PlanarImage pi = image;
  JPanel result = new JPanel( new BorderLayout() );
  if (pi != null) {
    SampleModel sm = pi.getSampleModel();
    ColorModel cm = pi.getColorModel();
```

viene poi calcolato il numero di righe che verranno mostrate, per strutturare un layout a griglia composto dalle due colonne previste e dal numero di righe necessarie. Nel caso il modello di colore non sia presente nell'immagine, il numero di righe sarà inferiore:

```
    int rowsCount = (cm != null) ? 7 : 4;
    JPanel contents = new JPanel( new GridLayout(
                                      rowsCount, 2) );
```

a questo punto si comincia ad aggiungere una serie di oggetti *JLabel* che contiene la descrizione dell'informazione ed il dato vero e proprio. La dimensione si ottiene da esempio direttamente dall'oggetto PlanarImage:

Fig. 2 Un esempio di ipotetico errore

```
    contents.add(new JLabel("Dimensioni"));
    contents.add(new JLabel( pi.getWidth()+"x"+
                               pi.getHeight()+" pixel"));
```

si ottengono da PlanarImage anche le informazioni sul tile. Un tile è un elemento separato di una immagine, una sua parte. In queste righe si estrae dunque l'informazione che indica quanti tile sono presenti nell'immagine:

```
    contents.add(new JLabel("Tile"));
    contents.add(new JLabel(pi.getTileWidth()+"x"+
     pi.getTileHeight()+" pixel"+" ("+pi.getNumXTiles()
```

NOTA

GESTIONE DEGLI ERRORI

Un elemento interessante è relativo alla gestione degli errori. Il metodo write() può generare una eccezione di tipo IOException, che identifica problemi di varia natura in fase di scrittura. Ad esempio si potrebbe aver selezionato un disco che non ha abbastanza spazio libero, oppure un percorso accessibile in sola lettura. L'errore viene intercettato dall'istruzione catch ed all'utente viene mostrato un messaggio. Questo viene presentato attraverso la classe JOptionPane, che dispone di diversi metodi. Uno di questi è *showMessageDialog()*, che mostra una finestra informativa.

Il primo parametro indica l'oggetto padre, il secondo il messaggio da riportare. È possibile indicare anche il titolo della finestra e la tipologia di segnalazione: tipo errore, messaggio, avvertimento, domanda, normale. La tipologia identifica una diversa icona.

```
+ "x"+pi.getNumYTiles()+" tile)"));
```

a questo punto è possibile estrarre informazioni sui campioni, compreso il tipo di dato utilizzato, che viene decodificato in funzione del valore dell'attributo dataType:

```
//informazioni sui campioni
contents.add(new JLabel("Numero di bande"));
contents.add(new JLabel(""+sm.getNumBands()));
contents.add(new JLabel("Tipo di dato"));
```

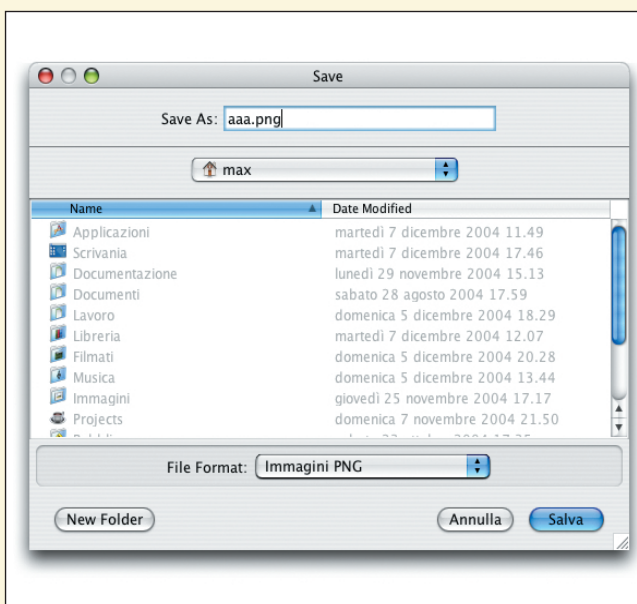
```
String tipo = "";
switch( sm.getDataType() ) {
    case DataBuffer.TYPE_BYTE:tipo = "byte";
        break;
    case DataBuffer.TYPE_SHORT:tipo = "short";
        break;
    case DataBuffer.TYPE_USHORT:tipo = "ushort";
        break;
    case DataBuffer.TYPE_INT:tipo = "int";
        break;
    case DataBuffer.TYPE_FLOAT:tipo = "float";
```

SALVARE LE IMMAGINI

GESTIRE LA VOCE DI MENU

```
saveMenuItem = new JMenuItem("Salva");
saveMenuItem.addActionListener( new ActionListener() {
  public void actionPerformed( ActionEvent e ) {
    salva();}}
);
```

1 **Aggiungere la voce "Salva" del menù installando un ActionListener che chiami il metodo salva(). Questo si occuperà di salvare l'immagine così come l'utente la vede a video.**

INVOCARE LA FINESTRA SALVATAGGIO

3 ***int result = fc.showSaveDialog(frame);***
Per fare in modo che Swing produca una finestra di salvataggio invece che di selezione file, è necessario chiamare il metodo showSaveDialog(). Il valore di ritorno funziona in modo similare all'apertura.

IMPLEMENTAZIONE FINESTRA SALVATAGGIO FILE

```
void salva() {
  JFileChooser fc = new JFileChooser();
  fc.setFileFilter( new FileFilter() {
    public boolean accept(File f) {
       String fileName = f.getName().toLowerCase();
       return fileName.endsWith(".png");}
    public String getDescription() {
      return "Immagini PNG"; }}
);
```

2 **Nel metodo salva() creare un JFileChooser, che sarà utilizzato per generare la finestra di salvataggio. Viene impostato un FileFilter che supporta il solo formato PNG, per maggiore semplicità. La descrizione del filtro è dunque semplicemente "Immagini PNG".**

SALVARE L'IMMAGINE

```
if (result == JFileChooser.APPROVE_OPTION) {
  File file = fc.getSelectedFile();
    try {
      ImageIO.write(imagePanel.image, "png", file );
    }
    catch (IOException e) {
       JOptionPane.showMessageDialog(frame, "Operazione non conclusa
          correttamente","JAI Photo", JOptionPane.ERROR_MESSAGE);
    }
  }
}
```

4 **Il metodo *salva()* si conclude con la gestione del salvataggio utilizzando le classi di JAI. per prima cosa viene estratto il nome del file selezionato, che viene poi passato al metodo *write()* della classe ImageIO. L'immagine da salvare viene presa invece dall'oggetto *ImagePanel*. Questo contiene l'immagine attualmente visualizzata a video.**

OPERAZIONI SINCRONE

5 **Si noti che l'operazione avviene in modo sincrono. L'interfaccia utente rimane infatti in attesa del completamento dell'operazione prima di restituire il controllo all'utente. Le altre operazioni, come l'apertura od i filtri grafici funzionavano invece in differita. Alla selezione del pulsante *Salva*, viene creato il file PNG relativo.**

```
                break;
            case DataBuffer.TYPE_DOUBLE:tipo = "double";
                break;
            case DataBuffer.TYPE_UNDEFINED:
            tipo = "undefined"; break; }
        contents.add(new JLabel(tipo));
```

anche le informazioni sul modello di colore vengono estratte in modo simile. Queste includono il numero di componenti di colore, il numero di bit per pixel e la tipologia di trasparenza:

```
        if (cm != null) {
            contents.add(new JLabel("Componenti colore"));
            contents.add(new JLabel(
                                    ""+cm.getNumComponents()));
            contents.add(new JLabel("Bit per pixel"));
            contents.add(new JLabel(""+cm.getPixelSize()));
            contents.add(new JLabel("Trasparenza"));
            String trasparenza = "";
            switch(cm.getTransparency()) {
              case Transparency.OPAQUE:
              trasparenza = "opaca"; break;
              case Transparency.BITMASK:
              trasparenza = "maschera di bit"; break;
              case Transparency.TRANSLUCENT:
              trasparenza = "translucente"; break; }
            contents.add(new JLabel(trasparenza));}
        contents.setBorder( new EmptyBorder( 5,5,5,5 ) );
        result.add( contents, BorderLayout.NORTH );}
    return new JScrollPane( result );
}
```

ma quando è possibile richiamare questo metodo? Ovviamente è necessario attendere che una immagine sia caricata in memoria. Si è scelto dunque di eseguire questa operazione nel metodo load(). Questo diventa quindi:

```
void load( final String filename ) {
  runner.run( new JAIOperation() {
    public PlanarImage run() {
      image = JAI.create("fileload", filename);
         originalImage = image;
         //informazioni sull'immagine
         if (infoPanel != null) {
           content.remove( infoPanel );}
         infoPanel = createInfoPanel();
         content.add( infoPanel, BorderLayout.EAST );
         return image;}
      public String getDescription() {
         return "loading " + filename;}
  });
}
```

subito dopo l'assegnazione ad originalImage è presente il nuovo codice. Per prima cosa questo controlla se esiste già un pannello di informazione

Fig. 3: *Dati di dettaglio di una immagine*

agganciato al pannello che contiene l'interfaccia utente dell'applicazione. In caso positivo lo rimuove. Poi crea il nuovo pannello, con le informazioni relative all'immagine appena caricata e lo aggiunge all'interfaccia utente.

CONCLUSIONI

Con questa puntata si conclude il nostro breve viaggio nelle API Java Advanced Imaging, un argomento interessante e che aggiungerà sicuramente molte frecce al vostro arco di programmatori Java.

Massimiliano Bigatti

FILE ED ICONE

Una classe di estensione di JFileChooser è *FileView*, che permette di associare a ciascun file una descrizione ed una icona, come di manipolarne il nome o di ritornare una descrizione dell'estensione.

LE INFORMAZIONI SULL'IMMAGINE

In un file di immagine non sono presenti solo i dati dei pixel, ma anche un profilo di colore. Inoltre, in funzione del numero di colori (256, 65535, TrueColor e così via) le strutture dati a supporto possono essere differenti. Ad esempio, per contenere un valore di colore nel range 0-255 è sufficiente un byte. Per un valore TrueColor ne servono tre, più uno per la trasparenza. Anche i dati delle immagini nascondono delle sorprese. Si pensi ai diversi formati di file supportati da JAI: GIF, PNG, JPEG, TIFF, BMP. Ciascuno di questi prevede un algoritmo di compressione specifico. Ad esempio, GIF, PNG e TIFF hanno algoritmi loseless. In questo caso i dati delle immagini sono compresse, e non c'è perdita di qualità. Il formato TIFF infatti è utilizzato, ad esempio, per i documenti acquisiti da scanner o inviati tramite fax, dove tutti i dati devono essere mantenuti. Il formato JPEG è invece lossy, prevede dunque una certa perdita di informazioni da "scambiare" con un minore spazio di memoria richiesto. Un formato loseless è ideale per i documenti, magari con tabelle e tanto testo; i formati lossy sono più indicati per le fotografie. Il motivo è semplice: la compressione lossy si nota meno su immagini quali ritratti, panorami od in generale con immagini prive di contorni demarcati. Tutti questi diversi formati si traducono con algoritmi di compressione/ decompressione diversi. Questi sono implementati in JAI e lo sviluppatore non se ne deve preoccupare. Ma questo si riflette anche sul modo in cui i dati dei pixel vengono memorizzati. Questi sono infatti suddivisi in "bande", ciascuna delle quali può contenere un componente di colore. Ad esempio, l'immagine in Figura 2 contiene tre bande, una per ciascun componente di colore (RGB). Inoltre, ciascuna banda contiene una serie di elementi di un specifico tipo. Quelli supportati da JAI sono: *byte; short; unsigned short; int; float; double;* ovviamente, una banda con dati di tipo byte potranno contenere valori da 0 a 255. Tutte le bande di una immagine sono racchiuse nel "modello dei sample".

Creiamo librerie personalizzate di tag JSP con i custom tag

Inventiamo nuovi Tag per JSP

Impareremo cosa sono, a che servono, come si creano, come si utilizzano e come si installano le librerie di custom tag in una Web Application

Utilizza questo spazio per le tue annotazioni

Esiste un problema comune che interessa le tecnologie basate su template page come JSP, ASP, PHP. Ovviamente stiamo parlando della “non separazione” della responsabilità delle pagine. Spesso infatti, ci si trova a dover gestire in un’unica pagina sia logica applicativa che logica di presentazione. Per alcuni sviluppatori questo tipo di programmazione risulta in prima istanza semplice e conveniente. La realtà è che in futuro il codice risulterà difficile da debuggare, poco leggibile e molto poco manutenibile. La strada da seguire è ovvia: separare il codice applicativo dalla logica di presentazione delle pagine. L’obiettivo di questo articolo è ambizioso. Rimuovere del tutto il codice Java dalla JSP, sostituendolo con dei tag JSP personalizzati creati ad hoc. I vantaggi saranno:

1) Separazione completa tra codice Java e codice HTML.

2) Codice leggibile, semplice ed elegante.

3) Riutilizzabilità del codice: i custom tag sono componenti riutilizzabili.

4) Supporto alla separazione dei ruoli in un team di sviluppo con conseguente aumento della produttività. Il web designer si occupa della formattazione delle pagine sfruttando tag HTML e custom tag. Lo sviluppatore Java può progettare, creare, documentare i custom tag.

I tag possono essere molto semplici, come gli *“empty tag”*, ma anche molto complessi, come gli *“iterative tag”* o i *“cooperative tag”*. Per esempio con un tag iterativo *(iterative tag)* ed un tag vuoto *(empty tag)*, sarà possibile utilizzare una sintassi del genere:

```
<table>
  ...
  <td align="left">
    <%-- Custom tag iterativo --%>
      <search_engine:iterateOverResults>
        <br>
          <%-- Custom tag empty --%>
       <search_engine:siteString/>
    <br>
  </search_engine:iterateOverResults>
</td>
```

Volendo si potrebbe addirittura creare un unico tag per creare la tabella dei risultati (!):

```
< search_engine:createTableResult />
```

Dopo aver letto i prossimi paragrafi saremo in grado di creare, configurare, ed utilizzare una libreria di tag personalizzati.

IL PRIMO CUSTOM TAG

I Custom Tag definiscono una tecnologia che estende il linguaggio JSP, con tag definiti dallo sviluppatore. L’idea di base è alquanto scontata: sono implementati mediante una classe Java (che estende la classe *TagSupport* del package *javax.servlet .jsp.tagext*), ma vengono invocati da una pagina JSP mediante un tag definito dallo sviluppatore. Vengono definiti come elementi di una libreria di tag, tramite un file XML detto *“Tag Library Descriptor”* (TLD). Inoltre, lo sviluppatore deve anche configurare il mapping tra la classe d’implementazione e il tag di invocazione, mediante il *TLD* e il *Deployment Descriptor* della Web Application.
I Custom Tag hanno una sintassi XML, e quindi seguono le regole di innesto dell’XML e sono case-sensitive. Esistono due tipologie di custom tag: gli *empty tag* (tag vuoti) e gli *standard tag* (contenenti un body).

Un tag vuoto ha una sintassi del tipo:

```
< prefisso:nome [attributo = “valore”]* />
```

mentre uno standard tag è definito nel seguente modo:

```
< prefisso:nome [attributo = “valore”]* />
  body
< /prefisso:nome />
```

Il prefisso serve per dare un namespace al tag, e viene definito nella stessa pagina dove si utilizzano i tag. Serve per non dare luogo ad ambiguità relativi ai nome dei tag. Come per ogni tag anche per i custom tag è possibile definire attributi. Come primo esempio, creeremo un empty tag ovvero un tag vuoto. In particolare creeremo un tag che mostra la data corrente formattata. Per intenderci il nostro tag dovrà sostituire il seguente scriptlet:

```
<%
  SimpleDateFormat sdf = new
                     SimpleDateFormat(“dd/MM/yyyy”);
  Date date = new Date();
  String dateString = sdf.format(date);
  out.print(dateString);
%>
```

o la contratta espressione equivalente:

```
<%=new SimpleDateFormat(
            “dd/MM/yyyy”).format(new Date()); %>
```

Ecco allora come dovremo scrivere il nostro tag:

```
<custom_tags:formatDate pattern=“dd/MM/yyyy” />
```

Dove:

- **custom_tags** è il prefisso che abbiamo scelto per la nostra libreria di tag.
- **formatDate** è il nome esplicativo del nostro tag.
- **pattern** è un attributo non obbligatorio del tag che serve per definire la tipologia di formattazione.

Per poter utilizzare tale tag in una pagina JSP, occorre ancora:

1) Scrivere la classe che viene detta *“tag handler”* (gestore del tag), che contiene il codice che deve essere eseguito.
2) Dichiarare un file *“Tag Library Descriptor”* (TLD), che descrive e dichiara l’esistenza del nostro tag.
3) Implementare il mapping del TLD con un certo identificativo all’interno del deployment descriptor *web.xml*.
4) Dichiarare all’interno della JSP l’utilizzo della libreria mediante una opportuna direttiva.

LA CLASSE TAG HANDLER

Una Tag Library è un componente web contenente un file *Tag Library Descriptor (TLD)*, e tutte le classi tag handler associate. In una classe tag handler sono visibili tutti gli oggetti impliciti della JSP. È quindi possibile accedere per esempio ad attributi della request o della session ed eseguire ogni istruzione che è possibile eseguire in una JSP. Come è possibile notare dal class diagram in **Figura 2**, una classe tag handler deve estendere la classe astratta TagSupport. *TagSupport* implementa l’interfaccia *Tag* ereditandone delle variabili statiche e dei metodi astratti. Questi ultimi vengono ridefiniti per avere un comportamento di default. Inoltre *TagSupport* possiede un’associazione verso *PageContext*, per poter accedere tra l’altro ad alcuni oggetti impliciti. *PageContext* rappresenta il “contesto della pagina” ovvero la JSP dove il custom tag viene inserito. La classe tag handler che lo sviluppatore deve creare, eredita in particolare i metodi *doStartTag()*, *doEndTag()* e *release()*. Questi ultimi risultano fondamentali perché rappresentano il modo mediante il quale l’application server gestisce il ciclo di vita di un custom tag.

Fig. 1: **Un Deployment Diagram per J2EE con custom tags**

Fig. 2: **class diagram semplificato della gerarchia di classi alla base di un tag handler**

Infatti, quando viene utilizzato un custom tag, l’application server

1) invoca per prima il metodo *doStartTag()*, e controlla quale costante ritorna.
2) Se questa coincide con la costante *EVAL_BODY_INCLUDE*, viene valutato il body interno al tag, e poi si passa al punto **3**). Se invece il metodo *doStartTag()* ritorna la costante *SKIP_BODY*, come nel caso di un empty tag, si passa direttamente al punto **4**).
3) Viene chiamato il metodo *doEndTag()*.
4) Viene invocato il metodo *release()*.

Andiamo quindi ad analizzare il codice del tag handler che formatta la data corrente. Trattandosi di un

SCRIPTLET

In una JSP si dice “scriptlet” un particolare tipo di tag che permette di scrivere codice Java libero, tramite la sintassi <% codice Java %>. I tag scriptlet possono essere alternati o innestati con altri tag JSP-HTML. Anche se questa caratteristica delle JSP è molto potente e invitante, comporta alcuni limiti.

OGGETTI IMPLICITI

All'interno di una pagina JSP, è possibile accedere ad alcuni oggetti senza che ci sia la necessità di istanziarli. Per tale motivo vengono detti "oggetti impliciti". In particolare gli oggetti impliciti sono 8: request, response, session, application, config, page, pageContext, out. A questi si aggiunge exception, che è accessibile solo nelle pagine definite come "Error Page".

DEPLOYMENT DESCRIPTOR

il Deployment Descriptor è un file XML che serve per istruire l'application server su come deve funzionare la web application. Al suo interno si trovano per esempio, parametri di inizializzazione statici, o mapping di url a servlet. Deve avere un nome preciso (web.xml) e si deve trovare nella cartella WEB-INF della applicazione.

empty tag, il metodo *doEndTag()* non verrà definito.

```
package com.cdsc.pub.customtags;
import …
public class FormatDateHandler
  extends TagSupport {
 private String pattern = "dd/MM/yyyy";
 public void setPattern(String pattern) {
  this.pattern = pattern; }
 public int doStartTag() throws JspException {
  JspWriter out = pageContext.getOut();
  try {
   if (pattern != null) {
    SimpleDateFormat sdf = null;
    try {
     sdf = new SimpleDateFormat(pattern); }
    catch (IllegalArgumentException ex) {
     sdf = new SimpleDateFormat("dd/MM/yyyy"); }
    String dateString = sdf.format(new Date());
    out.print(dateString); } }
  catch (IOException ioe) {
   throw new JspException(ioe); }
  return SKIP_BODY; }
 public void release() { pattern = "dd/MM/yyyy"; }
}
```

La nostra classe *FormatDateHandler* che gestisce l'empty tag *formatDate*, dichiara come variabile d'istanza, l'attributo *pattern* relativo all'attributo del tag. Il metodo *setPattern(String)* , servirà all'application server per settare la variabile *pattern* dopo aver letto il valore dell'attributo pattern del tag. Viene assegnato un valore di default alla variabile, trattandosi di attributo opzionale. Ciò significa che quando si utilizza il tag *formatDate*, non è obbligatorio specificare l'attributo pattern, se si vuole una formattazione di default. Il metodo *doStartTag()*, è il metodo più interessante. Esso recupera da *pageContext*, l'oggetto *out* di tipo *JspWriter*. Questo coincide con l'oggetto implicito *out* della JSP che rappresenta il contesto in cui viene eseguito il custom tag. Quindi, l'utilizzo del suoi metodi *print()* influenzeranno direttamente la response della JSP. Il metodo *doStartTag()* continua con la formattazione della data con il pattern settato (se è stato settato) e in caso di eccezione, utilizza il valore di default. Infine stampa tramite l'oggetto out la data formattata. Il metodo *release()*, resetta il valore dell'attributo opzionale pattern, al suo valore di default. Questo metodo viene invocato dall'application server come ultimo metodo dopo ogni utilizzo del tag. Ciò risulta fondamentale perché un custom tag, viene compilato come se facesse parte della JSP in cui viene chiamato, generandone una parte di codice. Sappiamo che una JSP viene alla fine compilata come servlet. Inoltre sappiamo che il ciclo di vita di una servlet, è gestito con un'unica istanza, con possibili problemi di concorrenza. Quindi anche un custom tag in pratica viene istanziato una sola volta, per ogni utilizzo. Se non esistesse il metodo *release()*, un'utenza che non specifica l'attributo pattern per il tag, finirebbe con l'usare il pattern che l'utente precedente aveva specificato!

Fig. 3: ***Relazioni tra file per la configurazione di un Custom Tag***

IL FILE TLD

Il file *Tag Library Descriptor (TLD)*, rappresenta il mezzo che l'application server utilizza per ricavare informazioni utili sulla libreria di custom tag che vuole utilizzare. Si tratta di un file XML molto semplice, di cui vediamo direttamente quello che abbiamo scritto per poter utilizzare il custom tag *formatDate*.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
  PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag
                                        Library 1.2//EN"
  "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_
                                              1_2.dtd">
<taglib>
 <tlib-version>1.2</tlib-version>
 <jsp-version>1.2</jsp-version>
 <short-name>IoProgrammo Custom Tags Library
 </short-name>
 <uri>http://www.claudiodesio.com/java/taglib</uri>
 <description></description>
```

Esempio di libreria di custom tag per ioProgrammo.

```
 <tag>
  <name>formatDate</name>
  <tag-class>com.cdsc.pub.customtags.FormatDateHandler
  </tag-class>
  <body-content>empty</body-content>
  <description></description>
```

Questo *empty tag* formatta la data corrente con un certo pattern.
L'attributo opzionale pattern permette di settare il tipo di formattazione della data.
Il pattern di default è *"dd/MM/yyyy"*.

```
   <attribute>
    <name>pattern</name>
    <required>false</required>
    <rtexprvalue>false</rtexprvalue>
  </attribute></tag>
</taglib>
```

Dopo le immancabili dichiarazioni di prologo XML, e di un DTD esterno, viene dichiarato l'elemento principale <*taglib*>. Di seguito le versioni sia delle specifiche TLD, che JSP, e uno <*short-name*>, che può essere utilizzato da eventuali strumenti come IDE ed application server. Fondamentale è la dichiarazione del tag <*uri*>, che assegna un identificatore alla libreria di tag. Questo sarà utilizzato dal deployment descriptor della web application *web.xml*, per mappare la libreria e renderla disponibile alle JSP. Per il tag <*description*> vale più o meno il discorso fatto per <*short-name*>. In questo file viene (per ora), dichiarato un unico custom tag, mediante il tag <*tag*> (il gioco di parole era inevitabile…). Vengono poi dichiarati il nome del tag *"formatDate"*, la sua classe tag handler *"FormatDateHandler"*, e la sua tipologia *"empty"*. I possibili valori per il <*body-content*> sono:

- **Empty:** nessun body-content.
- **JSP:** il tag può accettare codice JSP nel body.
- **tag-dependent:** il tag può accettare contenuto arbitrario nel body. In questo caso l'application server, non processerà il body ma lo passerà direttamente al tag handler.

Infine viene dichiarato anche l'attributo, che nel nostro caso è unico ma potrebbero anche essercene tanti. Per ogni attributo va dichiarato sempre il nome, l'obbligatorietà tramite il tag <*required*>, e il *"runtime expression value"* mediante il tag <*rtexprvalue*>. Quest'ultimo, se impostato a true permetterebbe di settare "al volo" il valore dell'attributo, anche con codice JSP dinamico. Per esempio, l'espressione:

```
<custom_tags:formatDate pattern="
                    <%= getLocalePattern() %>" />
```

setta il valore dell'attributo pattern mediante la chiamata ad un metodo invocato in una espressione JSP.

IL DEPLOYMENT DESCRIPTOR

L'ultimo passo che bisogna fare per utilizzare il nostro primo Custom Tag, è quello di configurare il file *web.xml* della nostra web application. Bisogna infatti far sapere all'application server che vogliamo utilizzare un nuova libreria tag. Basta come al solito un semplice tag inserito alla fine del file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems,
                    Inc.//DTD Web Application 2.3//EN"
          "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>custom_tags</display-name>
...
<taglib><taglib-uri>
          http://www.claudiodesio.com/java/taglib
    </taglib-uri><taglib-location>
          /WEB-INF/custom_tags.tld
    </taglib-location></taglib>
</web-app>
```

In tag *<taglib>* mappa l'identificativo uri, con la vera location del file TLD. A questo punto il nostro custom tag è pronto per essere utilizzato.

LA JSP

Siamo finalmente in grado di poter utilizzare il nostro primo custom tag. Il tutto si svolge con due semplici passi:

1) dichiarare l'utilizzo della libreria

2) utilizzare il custom tag.

Il punto 1) viene assolto mediante una direttiva JSP, chiamata proprio *"taglib"*. Segue la direttiva che ci permetterà di utilizzare il nostro *Custom Tag:*

```
<%@ taglib uri="http://www.claudiodesio.com/java
                    /taglib" prefix="custom_tags" %>
```

abbiamo dichiarato l'utilizzo della libreria di tag il cui identificatore uri è *"http://www.claudiodesio.com/java/taglib"*. Abbiamo anche dichiarato mediante l'attributo prefix che in questa pagina utilizzeremo il namespace *"custom_tags"*, per riferirci ai tag della nostra libreria. Se volessimo utilizzare in un'altra JSP, la libreria, potremmo tranquillamente utilizzare un altro namespace. Infine non ci rimane che utilizzare il nostro tag, per esempio nel seguente modo:

```
<custom_tags:formatDate pattern="yyyy/MM/dd" />
```

CONCLUSIONI

Abbiamo cercato di spiegare cosa è, come si crea e come si utilizza un custom tag. Abbiamo visto i suoi vantaggi e i suoi svantaggi. Poi abbiamo creato, configurato, ed utilizzato un semplice custom tag vuoto. Il processo di sviluppo ha richiesto la scrittura di una classe il tag handler, di un file TLD e delle modifiche al web.xml.

Claudio De Sio Cesari

ESPRESSIONI

In una JSP si dice "espressione" un particolare tipo di tag che permette di scrivere stringhe direttamente nella response HTTP, tramite la sintassi <%= espressione stampabile %>. Con espressione stampabile intendiamo una qualsiasi istruzione che possa essere stampata, come un intero, un oggetto, un metodo che non abbia come tipo di ritorno void. Un'espressione produce lo stesso output del seguente scriptlet che stampa mediante l'oggetto out:

```
<% out.print(
        espressione); %>.
```

L'AUTORE

Claudio De Sio, è un consulente free-lance che si occupa di tecnologia Java, analisi, progettazione e architetture object oriented. Laureato in matematica, dal 1999 collabora con Sun Educational Services (come docente e mentore), ed altri importanti aziende dell'IT. Sul suo sito internet http://www.claudiodesio.com, ha pubblicato diverse risorse didattiche gratuite, relative agli ambiti dove è specializzato.

La prima applicazione con FlashLite 1.1

Macromedia FlashLite e i Cellulari

La nuova versione FlashLite del player di Macromedia permette di visualizzare contenuti multimediali all'interno dei cellulari che lo supportano

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste

ActionScript 1.0 ed una discreta padronanza nello sviluppo di applicazioni Flash multi-piattaforma

Software

Flash Mx 2004 Porfessional

Impegno

Tempo di realizzazione

FlashLite è una versione molto leggera del Flash Player creata da Macromedia per essere utilizzata all'interno di dispositivi poco potenti e con poca memoria a disposizione come i telefoni cellulari. La versione attualmente disponibile è la 1.1 che, rispetto alla precedente release, mette a disposizione un set più ampio di comandi, un *CDK (Content Development Kit)* molto curato e ricco sia di materiale che di strumenti di sviluppo, il supporto SVG-T, la possibilità di connettere le proprie applicazione a fonti dati esterne e un elevato livello di interazione con il dispositivo su cui è installato del quale può recuperare informazioni come la potenza del segnale, il livello della batteria, ecc. e del quale può sfruttare funzionalità come l'invio di SMS.
La prima impressione che si ha leggendo le specifiche di questo player che basa il suo funzionamento sulla versione quattro del Flash Player può essere poco positiva e assolutamente non entusiasmante in particolar modo per chi in passato si è scontrato con i limiti dell'ActionScript disponibile in Flash 4.
Ma come spesso accade la prima impressione non rispecchia totalmente la realtà e, attraverso un attento esame della documentazione, ci si rende conto che, al di là dei limiti evidenti del vecchio player, ci sono una serie di comandi, comunemente denominati *FSCommand2*, che ci permettono di creare applicazioni sofisticate e raffinate per cellulari.

COME INIZIARE

1) È necessario avere installato sul computer Macromedia Flash Mx 7.2. o superiore

2) Scaricare e installare il kit di sviluppo all'indirizzo http://www.macromedia.com/devnet/flashlite.html.
Per testare le vostre applicazioni dovete scrivere un e-mail alla divisione Mobile di Macromedia mobiledeveloper@macromedia.com, fornire il numero IMEI (si ottiene digitando sulla tastiera la combinazione di caratteri *#06#) del vostro cellulare e il modello che, al momento, deve essere uno di quelli riportati nella lista ufficiale presente sul sito Macromedia all'indirizzo http://www.macromedia.com/mobile/supported_devices/.
Otterrete un file .sis installabile sul cellulare e potrete così verificare il corretto funzionamento dell'applicazione oltre che nell'ambiente di authoring di Flash anche su un dispositivo mobile.
Al momento della stesura di questo articolo Macromedia ha ufficialmente annunciato che entro i primi mesi del 2005 verranno inseriti nella lista dei telefoni cellulari che supportano FlashLite nuovi modelli.

TECNICHE DI SVILUPPO

Le applicazioni per cellulare, per mantenere un peso molto ridotto, devono essere scritte utilizzando la sintassi e le funzioni ActionScript presenti in Flash 4 combinate con alcune funzionalità presenti in Flash 5. L'utilizzo di questa versione "ibrida" di codice non presenta particolari difficoltà, è necessario solo prestare una maggiore attenzione e tenere presenti alcuni casi particolari. Se all'interno del codice contenuto nell'applicazione volete riferivi alla timeline principale potete utilizzare la slash sintax di Flash 4 combinata con il punto o le classiche keyword *ActionScript _level0* o *_root* introdotte nella versione 5 di Flash. Vediamo ad esempio come riferirci ad un clip filmato presente sullo stage dalla timeline principale al quale abbiamo assegnato il nome istanza *square_mc* e che a sua volta è contenuto in un altro *MovieClip* il cui nome istanza è *ball_mc.*
La sintassi corretta è

```
tellTarget("/ball_mc/square_mc");
tellTarget("_level0/ball_mc/square_mc");
```

Per accedere invece dal clip filmato *square_mc* al vostro stage le due forme sintattiche corrette che si

possono utilizzare sono

```
tellTarget("../../square_mc");
tellTarget("_level0");
```

LE VARIABILI

Uno degli strumenti fondamentali utilizzati in programmazione solo le variabili, dichiariamo la variabile a sulla timeline principale e vediamo quale sintassi è possibile utilizzare per accedere ad essa rispettivamente dal clip filmato *square_mc* e dal clip *ball_mc*

```
trace(/:a)
trace(_root.a);
```

L'utilizzo combinato di *slash (/)* e *colons (:)* o della keyword *_root* permette di recuperare il valore memorizzato nella variabile *a*.

LOOP ED EVENTI

Ai tempi della versione quattro, versione a cui facciamo riferimento quando sviluppiamo per FlashLite, i *MovieClip events* non erano supportati in alcun modo. Per eseguire del codice contenuto all'interno della timeline dobbiamo creare dei loop combinati con istruzioni condizionali all'interno della linea temporale del filmato. Vediamo ad esempio come intercettare la pressione della freccia sinistra della tastiera e fare in modo che, una volta rilevato l'evento, il clip filmato square_mc si sposti lungo l'asse delle *_x* fino ad una precisa posizione. Aggiungete un livello sulla timeline principale del filmato e inserite al suo interno un pulsante che potete prendere dalle librerie comuni o creare voi stessi. Il codice da associare al pulsante deve rilevare la pressione di un particolare tasto e fare in modo che venga eseguita una porzione di codice finché non è soddisfatta una particolare condizione. In parole povere non farete altro che mettere in stop la timeline, rilevare l'attuale posizione sullo stage del clip filmato e avviarne successivamente la riproduzione in modo che, nei fotogrammi successivi, venga eseguito un controllo sulla posizione e la ripetizione di un blocco di azioni. Il pulsante rileva quindi un evento e manda in play la timeline principale

```
on(keyPress "<left>")
{  play(); }
```

Per gestire lo spostamento sull'asse delle *_x* è sufficiente memorizzare in una variabile il valore della proprietà *_x* del clip filmato *square_mc* e incrementarne il valore. Recuperiamo la posizione e blocchiamo l'avanzamento della testina di riproduzione

```
pos = getProperty("square_mc" , _x);
stop();
```

Ora per far in modo che il clip si sposti lungo l'asse delle *x* aggiungete altri due fotogrammi alla timeline principale ed un livello etichette che sfrutterete per mantenere ordinato il vostro *.fla*. Assegnate al secondo frame l'etichetta "*process*" e al suo interno inserite lo script

```
pos += 5;
_level0.square_mc._x = pos;
```

Nell'ultimo fotogramma verificate il valore memorizzato nella variabile *pos* e fate in modo che venga eseguito nuovamente il frame "*process*" finché il clip filmato *square_mc* non ha raggiunto un determinato punto sullo stage

```
if(pos > 300){
  gotoAndStop("start"); }else
{ gotoAndPlay("process"); }
```

GESTIRE LA TASTIERA

Quando si sviluppa per telefoni cellulari si deve tenere sempre presente che l'unico strumento a disposizione degli utenti per interagire con l'applicazione è la tastiera dell'apparecchio o, nei casi più fortunati, il joystick integrato. Per questo motivo è opportuno che inseriate tutti gli eventi da rilevare all'interno di un unico pulsante posizionato fuori dallo stage che richiama diversi blocchi di istruzioni. Gli eventi che si possono rilevare sono:

```
- keyPress "<Up>"
- keyPress "<Down>"
- keyPress "<Left>"
- keyPress "<Right>"
- keyPress "<Enter>"
- keyPress "0" (sono accettati valori compresi tra 0 e 9)
- keyPress "*"
- keyPress "#"
- keyPress "<PageDown>"
- keyPress "<PageUp>"
```

RICHIAMARE BLOCCHI DI CODICE

Per richiamare un blocco di codice quando si lavora su applicazioni progettate per FlashLite si deve utilizzare il comando *call()*, considerato ormai obsole-

Fig. 1: **Verifica dell'installazione degli strumenti di sviluppo e pubblicazione**

NOTA

Quando ci troviamo in procinto di utilizzare Flash per un'applicazione distribuita su dispositivi mobili dalle capacità così ridotte come i telefoni cellulari, dobbiamo pensarla e strutturarla in modo tale che risulti essere il più leggera, al massimo 100 Kb, e che non impegni troppo né la cpu del dispositivo né la memoria a disposizione. Per questo motivo è molto importante porre una adeguata cura nell'ottimizzazione delle immagini che volete usare, evitare transizioni con effetti molto complessi, tenere presente che la navigazione avviene solo da tastiera ed è necessario considerare quali formati audio sono supportati dai telefoni cellulari.

to, che vi permette di simulare le function introdotte dalla versione 5 di Flash. Aggiungete un nuovo fotogramma alla timeline principale, inserite una semplice azione di *trace()* al suo interno e definite in corrispondenza di questo frame l'etichetta hello che sarà utilizzata come argomento della funzione *call()*. Con il comando *call()* infatti non fate altro che forzare l'esecuzione del codice contenuto in un fotogramma al quale è stata associata una etichetta. Per testare il corretto funzionamento aggiungete un pulsante sullo *stage* e, rilevando la pressione della freccia destra, richiamate il codice contenuto nel frame *hello*

```
on(keyPress "<right>")
{
    call("hello")
}
```

GESTIRE FILMATI DIFFERENTI

Per gestire la comunicazione tra diversi clip filmato all'interno della vostra applicazione è necessario utilizzare un'altra azione obsoleta rimasta ormai storica: *tellTarget()*. La sua sintassi è molto semplice ed è questo il comando necessario per fare in modo ad esempio che la timeline di un clip filmato si posizioni ad un determinato fotogramma

```
tellTarget("nomeclip")
{
    gotoAndStop(2);
}
```

L'azione *getURL* è supportata in FlashLite anche se può essere eseguita solo se associata alla pressione dei tasti numerici *(0 – 9)*, dell'asterisco *(*)* o del cancelletto *(#)*.
I protocolli supportati sono quelli standard fatta eccezione per il protocollo tel che consente di effettuare

IL DISEGNO DELL'APPLICAZIONE

I componenti possono essere posizionati nel layer contents diviso in due parti tramite un blank keyframe. Gli elementi contenuti nei frame dall'1 al 4 saranno gestiti dal codice contenuto nel box "Le azioni". Gli elementi dal frame 5 al 9 vengono gestiti dal codice contenuto nel box "Gestire la navigazione". Un bottone nascosto catturerà l'input dal tastierino numerico.

FACCIAMO VIBRARE IL CELLULARE

UN NUOVO PROGETTO

1 **Dal Menu *File* di Flash, selezioniamo *Nuovo* e dalla tabsheet *Template* selezioniamo "*Mobile Devices*" e di seguito un modello di telefono adatto a ricevere la nostra applicazione. Per il nostro esempio abbiamo scelto un Nokia 7650.**

AGGIUNGERE I LIVELLI

2 **Cliccando con il mouse sul tasto aggiungere i livelli *Actions, Labels, Up Bar, Navigation, Down Bar, Content* esattamente nell'ordine che vi abbiamo specificato. Per rinominare un livello è sufficiente un doppio click sulla sua etichetta.**

DIVIDIAMO LA TIMELINE

3 **Portatevi con il mouse suò 4 frame del layer *content*, utilizzate il tasto sinistro del mouse e scegliete *insert Blank Keyframe,* di seguito portatevi sul nono frame dello stesso livello e utilizzando il tasto sinistro del mouse scegliete *insert frame*.**

LE AZIONI

Questo codice attribuisce alle varie label il contenuto appropriato. Il contenuto viene ricavato associando alle variabili il risultato prelevato dal comando FSCommand corretto

```
// imposto full screen
fscommand2("FullScreen", true);
// imposto il valore della proprietà quality
fscommand2("SetQuality", "high");
// disabilito il bordo giallo visibile se un elemento
// dell'interfaccia riceve il focus
_focusrect = false;
web = _capLoadData;
memo = FSCommand2("GetTotalPlayerMemory");
data = FSCommand2("GetLocaleShortDate");
language = FSCommand2( "GetLanguage",
                                        "language");
bat = FSCommand2( "GetBatteryLevel");
sig = FSCommand2( "GetMaxSignalLevel");
// segnale e batteria
tellTarget ("battLevel_mc") {
            gotoAndStop(/:bat);
}
tellTarget ("sigLevel_mc") {
            gotoAndStop(/:bat);
}

// grafica per indicare la connessione disponibile
if(web == 1){
      tellTarget ("web_mc") {
            gotoAndStop("ok");
}
}else{
      tellTarget ("web_mc") {
            gotoAndStop("ko");
      }
}
stop();
```

chiamate a numeri telefonici direttamente dalle vostre applicazioni

```
getURL("tel: 117");
```

CARICARE DATI ESTERNI

A seconda delle caratteristiche del dispositivo su cui viene visualizzata la vostra applicazione e dei supporti offerti è possibile caricare dati esterni con i comandi *loadVariables* e *loadVariablesNum* o visualizzare altri *swf* con le azioni *loadMovie* e *loadMovieNum*.
Per caricare in FlashLite dati provenienti da pagine server side è necessario mantenere nei risultati generati dalla pagina il formato *variabile=valore* che solitamente trovate all'interno dei file *.txt* caricati in Flash.
Nonostante FlashLite supporti il formato *T-SVG*, al suo interno non è presente nessun par-

GESTIRE LA NAVIGAZIONE

Al primo frame associato al bottone nascosto inseriamo i comandi che ci servono per dare il via all'esecuzione dell'applicazione. L'applicazione arriverà al frame 5 dove si bloccherà in attesa dell'input

```
on(keyPress "<Enter>"){
      play();
```

menu_mc è un filmato che contiene il disegno dei singoli rettangoli che simulano il menù. right_mc, left_mc, up_mc e down_mc sono rispettivamente invece delle istanze delle frecce che simulano il joystick

```
on (keyPress "<Right>") {
      tellTarget ("menu_mc") {
            nextFrame();
      }
      tellTarget("right_mc"){
            gotoAndPlay("on");
      }
}
on (keyPress "<Left>") {
      tellTarget ("menu_mc") {
            prevFrame();
      }
      tellTarget("left_mc"){
            gotoAndPlay("on");
      }
```

LE AZIONI

4 **Con la stessa tecnica vista nel punto 3, dividete il layer *actions* inserendo un blank keyframe al frame 4 ed un frame al punto 9. Nel primo frame del livello actions inserite il codice come qui di fianco digitandolo nella finestra *Actions* come in figura.**

VIBRAZIONE

```
fscommand2("SetSoftKeys", "left", "right");

vStartVib = FSCommand2("StartVibrate",
                                2000, 2000, 2);

stop();
```

5 **Al frame 5 dell'applicazione inserite il codice come sopra. Noterete che se il filmato dovesse raggiungere questo frame il telefono inizierà a vibrare. Viene inoltre abilitato il tastierino all'intercettazione della pressione dei tasti sinistro e destro.**

GESTIRE LA NAVIGAZIONE

6 **Nello stage content, aggiungiamo un tasto nascosto, cliccando sul tasto aggiungeremo al primo frame l'azione come al box *"Le azioni"* e al secondo frame l'azione come al box *"Gestire la navigazione"*, questo ci consentirà di navigare fra le varie scene.**

ser XML, quindi, al momento, non potete caricare e gestire dati XML nelle applicazioni.

VARIABILI D'AMBIENTE

FlashLite mette a disposizione una serie di variabili che possono consentirvi di gestire in maniera differente i controlli e le funzionalità previste in fase di sviluppo per la vostra applicazione. La variabile *_capEMail* vi consente di rilevare se il player in cui è visualizzata la vostra applicazione è in grado di inviare posta elettronica attrverso il comando *getURL()*. Se la funzionalità è supportata il valore assunto da *_ capEMail* è 1, in caso contrario *undefined*. Le variabili *_capMMS* e *_capSMS*, in maniera del tutto simile a *_capEMail*, possono aiutarvi a stabilire se, attraverso il telefono cellulare che sta utilizzando la vostra applicazione, è possibile inviare MMS e/o SMS attraverso l'azione *getURL()*. Tra le varie possibilità offerte da FlashLite spicca la capacità di rilevare se il dispositivo è in grado di riprodurre suoni in streaming attraverso i valori restituiti dalla variabile *_capStreamSound* e se può gestire alcuni particolari formati audio. La variabile *_capMFi* vi indica sei il dispositivo è in grado di riprodurre suoni in formato MFi. I valori che può assumere sono 1 se il formato è supportato e undefined qualora il telefonino non sia in grado di gestire questo tipo di dati. Le variabili *_capMIDI* e *_capSMAF* funzionano in maniera analoga e vi permettono di rilevare se è possibile gestire attraverso il dispositivo il formato audio MIDI e/0 SMAF. Uno degli aspetti fondamentali che può decretare il successo di un'applicazione per cellulari è la sua capacità di aggiornare i dati in essa contenuti. Purtroppo non tutti i dispositivi offrono questo supporto e, utilizzando la variabile *_capLoadData*, preposta proprio al compito di rilevare la possibilità di caricare dati esterni, potete gestire in maniera differente la vostra applicazione.

Fig. 2: **Associazione di un file MIDI esterno**

GESTIONE DELL'AUDIO

La riproduzione di file audio è molto limitata nei telefoni cellulari e, anche le suonerie più accattivanti, devono sempre scontrarsi con la scarsa memoria a disposizione e le basse performance dei processori utilizzati in questi dispositivi. Per utilizzare un suono nelle vostre applicazioni è necessario quindi utilizzare formati audio come il MIDI che non possono essere gestiti direttamente nell'ambiente di sviluppo di Flash.

Se volete associare ad un determinato evento nella vostra applicazione un file .mid dovete importare in Flash un qualsiasi file audio in uno dei formati supportati ed eseguire un collegamento al .mid che rimane esterno ma che verrà incluso in fase di compilazione nel file .swf attivando con il tasto destro del mouse le proprietà del file audio importato in Flash.

COMANDI FSCOMMAND2

Le funzioni FSCommand2() non sono supportate all'interno dell'ambiente di authoring ma possono accettare argomenti, restituiscono un valore e vengono eseguite immediatamente. Attraverso queste nuove funzioni è possibile sia interagire con il telefono cellulare, sia eseguire un check delle sue funzionalità, sia gestire il caricamento dei dati esterni rilevando i problemi di connessione e/o il completo scaricamento dei dati. Vediamo di chiarire questo concetto attraverso alcuni semplici esempi. L'azione FSCommand2 ("GetNetworkRequestStatus"); restituisce un valore numerico compreso tra 0 e 8 e ci permette di rilevare gli step che il telefono cellulare esegue per connettersi ad una fonte dati esterna. Attraverso l'istruzione FSCommand2 ("StartVibrate", time_on, time_off, repeat) è possibile avviare, se il telefono su cui è visualizzata l'applicazione lo consente, la vibrazione scegliendone la durata e il numero di volte che deve essere eseguita. Il livello di interazione con il dispositivo mobile al quale si può arrivare attraverso in nuovi FSCommand è veramente elevato. L'azione FSCommand2 ("SetSoftkeys", var1, var2); consente di associare del testo alla pressione della freccia destra o sinistra del joistyck integrato nel cellulare mentre attraverso il comando FSCommand ("Launch", "applicazione"); potete aprire ed eseguire altri programmi installati nel dispositivo.

DEBUG DELL'APPLICAZIONE

Per testare l'applicazione potete avvalervi dell'ambiente di authoring di Flash ricordandovi che, in modalità prova filmato, per provare la navigazione da tastiera, è consigliabile disabilitare le shortcut attivando il menu *Control* e selezionando la voce *Disable ShortCuts* e che la finestra di output, quando si utilizza il porfilo di pubblicazione FlashLite1.1 riporta molti messaggi dedicati proprio alla verifica del corretto funzionamento del file *.swf* generato. Per un debug accurato è consigliabile provare periodicamente il file generato all'interno del telefono cellulare per cui state sviluppando l'applicazione trasferendolo con i software dedicati al trasferimento dati.

Giorgio Natili

La tecnologia di Microsoft per lo sviluppo di applicazioni Web

Programmare in ASP.NET

Primi passi per imparare a creare siti Web utilizzando la tecnologia ASP.NET di Microsoft. Diremo qualcosa sul compilatore e faremo un esempio d'uso

Come molti di voi sapranno, ASP.NET è la tecnologia proposta da Microsoft per lo sviluppo di pagine Web. In questo articolo ci preoccuperemo di sviluppare una prima pagina Web e diremo qualcosa sul funzionamento interno della tecnologia .NET, soffermandoci nei prossimi articoli sulle caratteristiche avanzate.

DI COSA ABBIAMO BISOGNO?

Prima di tutto del .NET Framework SDK, si può scaricare da *http://www.asp.net/*.
È possibile realizzare una pagina Web ASP .NET anche con il notepad. Tuttavia esistono ambienti che semplificano moltissimo la vita a un programmatore.

COS'È IL .NET FRAMEWORK

Alla base di tutto c'è il .NET Framework, un insieme di applicazioni e servizi che forniscono l'infrastruttura di base per lo sviluppo e l'esecuzione di applicazioni. Il .NET Framework, nella sua attuale versione 1.1, è limitato ai sistemi operativi Windows 9x, NT, 2000, XP e 2003. Per quanto riguarda la parte web, il supporto parte da Windows 2000 in su. Le versioni attualmente disponibili sono la 1.0 e la 1.1, quest'ultima già installa di default su Windows Server 2003. Il .NET Framework è soprattutto un insieme di classi unificate, che permette la creazione di applicazioni multi piattaforma sfruttando una sola volta la conoscenza acquisita nell'uso degli oggetti di cui è composto. Attraverso alcuni add-on è possibile sfruttare il .NET Framework per la creazione di qualsiasi tipo di applicazione:

- **web, con ASP.NET;**
- **Windows, con le cosiddette WinForms;**
- **Web Services;**
- **web per mobile device, ovvero applicazioni web pensate per palmari o telefoni cellulari, sfruttando ASP.NET**
- **per mobile device, in maniera nativa sfruttando il .NET Compact Framework, una versione ridotta in grado di funziona su palmari dotati di Windows Mobile 2003 o su telefoni cellulari Smartphone 2003.**

Come si vede la copertura delle diverse necessità che uno sviluppatore (e quindi un'azienda che ha bisogno di una soluzione) è garantita. Guardando un attimo il framework dal punto di vista del suo funzionamento, si basa su un compilatore JIT (Just in Time), che si occupa di compilare al volo un linguaggio particolare, chiamato *MSIL (Microsoft Intermediate Language)*.
Tutti i compilatori, infatti, producono codice MSIL, che poi a sua volta viene compilato ed eseguito dal JIT, dopo aver passato una serie di ottimizzazioni specifiche per ogni architettura hardware.
Il risultato di questo procedimento è che l'eseguibile generato è stato cucito addosso al sistema in cui andrà a girare e che viene eseguito all'interno di una sandbox, ovvero un contenitore che isola l'esecuzione di questo codice dal sistema operativo nel quale gira.
Grazie alle funzionalità di *CAS (Code Access Security)* è possibile anche definire politiche di accesso a determinate risorse da parte di programmi o classi, cosa che garantisce un altro grado di configurabilità.
Inoltre le applicazioni che girano all'interno del .NET Framework sono protette da buffer overrun, cosa che garantisce di essere immuni ad una buona parte dei problemi di security di cui soffrono le applicazioni tradizionali.

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
HTML, ASP.NET

Software
Microsoft .NET Framework 1.0 o successivi

Impegno

Tempo di realizzazione

L'ambiente principe è sicuramente il Microsoft Visual Studio.NET, tuttavia ne esistono anche altri. Scaricando il .NET Microsoft SDK avremo a disposizione tutti gli strumenti quali compilatori a riga di comando etc. utili se decisiamo di non utilizzare l'MS Visual Studio .NET.
L'alternativa economica a Visual Studio .NET è ASP .NET Web Matrix scaricabile da *http://www.asp.net/Default.aspx?tabindex=0&tabid=1*.
La differenza tra VS.NET 2003 e Web Matrix sta nella quantità di *facilities* messe a disposizione dai due ambienti. Nel primo caso si ha a disposizione un'IDE RAD, *Rapid Application Development*, che ha molte analogie con un programma per disegnare, ovvero consente di "disegnare" le proprie pagine trascinando gli elementi da una toolbox su un contenitore. Inoltre l'ambiente mette a disposizione l'intellisense che consente il completamento automatico del codice e molto altro ancora. ASP.NET Web Matrix è più indicato per gli hobbisti o per tutti coloro che non possono o vogliono investire nell'acquisto di un ambiente di sviluppo più avanzato, si tratta di un progetto del team di sviluppo di ASP.NET, che ha creato un editor molto semplice, con alcune caratteristiche minime per lo sviluppo visuale, ma senza Intellisense né supporto al code behind.
Il suo punto di forza è nella estrema semplicità e nella possibilità di provarlo senza dover acquistare nessuna licenza, dato che è gratuito.
Se avete intenzione di sviluppare siti Web che facciano accesso a database, è utile anche MSDE, una versione particolare di SQL Server, che si può scaricare da *http://www.asp.net/msde/* ed il cui utilizzo è gratuito con ASP .NET/Web Matrix. Permette di testare le proprie soluzioni basate su SQL Server anche in locale, con estrema facilità.
Manca di un equivalente dell'Enterprise Manager, che si può comunque sostituire egregiamente con l'italianissimo DbaMgr che si trova all'indirizzo *http://www.asql.biz/DbaMgr.shtm*

I LINGUAGGI DEL .NET FRAMEWORK

Il .NET Framework supporta principalmente due linguaggi, C# e VB.NET: http://msdn.microsoft.com/vcsharp/ http://msdn.microsoft.com/vbasic/
È però possibile sfruttarne anche altri, come Python, Delphi, Cobol, etc. Una lista completa si può trovare su http://www.asp.net/Default.aspx?tabindex=7&tabid=40

E PER PROVARE LE NOSTRE PAGINE?

Per far girare le applicazioni è necessario dotarsi di Windows 2000 Server o Windows Server 2003. Nel primo caso si ha IIS 5, nel secondo IIS 6 che è certamente la scelta migliore per quanto riguarda la stabilità generale del sistema.ASP.NET non funziona su Windows NT 4 o Windows 9x/ME, mentre si può usare Windows 2000 Professional o XP per lo sviluppo.
Nel caso di XP Home, dato che non è presente un web server, si può ripiegare su Cassini, scaricabile all'indirizzo *http://www.asp.net/Projects/Cassini/Download/* che è un web server molto leggero scritto in C# ed è ideale anche su XP Pro qualora non si possa usare IIS.

FACCIAMO UNA PROVA

Se avete installato Cassini o IIS, aprite il note-

WEB MATRIX IN PRATICA

Vi proponiamo un brevissimo esempio per creare una prima pagina ASP.NET. In questo breve esempio noterete già che abbiamo fatto uso di una programmazione a "eventi". Inoltre è facile intuire dall'esempio cosa sia una proprietà e cosa sia un metodo. Di questi aspetti torneremo a parlare in modo approfondito sulle nostre riviste negli altri articoli del corso. Per eseguire l'esempio, è sufficiente salvare la pagina nella root del web server che state usando e poi puntare il browser a *http://localhost/vostroesempio.aspx*

SELEZIONIAMO IL LINGUAGGIO

1 **All'avvio di Web Matrix vi verrà chiesto di indicare un nome e un percorso per il vostro file *.aspx*. È inoltre molto importante scegliere quale linguaggio userete per lo svilupp, fra Visual Basic, c# e j#. Per la nostra prova useremo VB.**

AGGIUNGIAMO UN'IMMAGINE

2 **Trasciniamo dalla toolboox sulla sinistra il componente "*Image*" nella pagina, avendo cura di settare nella tabsheet delle proprietà in basso a destra il percorso dell'immagine. Chiaramente deve trattarsi di un percorso assoluto e non relativo al nostro HD.**

pad e digitate dentro le seguenti righe:

```
<%@ Page Language="VB" %>
<SCRIPT runat="server">
Sub Button1_Click(sender As Object, e As EventArgs)
  label1.text="Hello World"
 End Sub
</script>
<html>
<head></head>
<body>
<form runat="server">
<asp:Label id="Label1" runat="server">
                                        Label</asp:Label>
<br>
<asp:Button id="Button1" onclick="Button1_Click"
                  runat="server" Text="Premimi!" />
</form>
</body>
</html>
```

Salvate la pagina come *prova.aspx* nella root del web server e puntate il vostro browser su *http://localhost/prova.aspx* qualche secondo dopo vedrete apparire una pagina con un bottone e un'etichetta.
Premendo il bottone l'etichetta cambia in *"Hello World"*.

UNA CLASSE PER OGNI NECESSITÀ

Il .NET Framework si basa su un'insieme di classi di base che forniscono tutte le funzionalità di cui possiamo aver bisogno. Il vantaggio di queste classi è che non necessitano di registrazione, fatta eccezione di quelle nella GAC (Global Assembly Cache), che si vogliano rendere disponibili a tutte le applicazioni di uno stesso server senza avere diverse copie sparse sul disco fisso. Poter distribuire liberamente senza registrazioni classi personalizzate, favorisce di molto il riutilizzo del codice, ma anche e soprattutto permette di scrivere, a tutti, applicazioni 3-tier (a 3 livelli), in grado di garantire una sana e giusta separazione tra i diversi strati (data, business, presentation layer). Gli oggetti sono organizzati all'interno di Namespace, ovvero di identificatori. Possiamo vederli un po' come indirizzi, che ci fanno capire una classe a quale funzionalità, a livello logico, assolve. Esiste per esempio il namespace System.Data, al cui interno ci sono varie classi responsabili per l'accesso ai dati. Le classi una volta compilate vengono anche chiamate Assembly e, come già detto, sono scritte in MSIL dai vari compilatori C# o VB.NET. Questo vuol dire che il codice sarà visibile in chiaro (in MSIL appunto) e dunque è possibile, senza troppo sforzo, generare il codice C# o VB.NET corrispondente. Esistono degli obfuscator (offuscatori) che "criptano" il codice, rendendo impossibile l'operazione di decompilazione ad utenti sprovveduti, ma ovviamente ci sono tools che permettono, ad utenti esperti, di recuperare anche parte di codice offuscato. All'interno di una classe possiamo trovare, come al solito, proprietà, metodi ed eventi. È in questo modo che noi possiamo utilizzare l'oggetto per le nostre applicazioni, valorizzando proprietà, eseguendo le funzioni contenute nei metodi e scatenando gli eventi associati.
All'interno del .NET Framework esistono classi per tutte le necessità: dall'accesso ai socket alla crittografia, all'accesso a fonti dati o file XML, alla gestione del file system. Molto spesso le classi di terze parti non sono altro che un sistema per estendere (o arricchire) funzionalità che in realtà sono già presenti nel framework.

AGGIUNGIAMO UN CALENDARIO E UNA LABEL

3 **Dala toolbox degli strumenti trasciniamo sulla pagina un calendario e una label, avendo cura di eliminare ogni valore dalla proprietà *text* della label. Se siete dei perfezionisti potreste volere usare una tabella per allineare i vari componenti**

AGGIUNGIAMO IL CODICE

```
Sub Calendar1_SelectionChanged(sender As
                        Object, e As EventArgs)
    label1.text="Hai selezionato "
                        +calendar1.SelectedDate
End Sub
```

4 **Cliccate due volte sul calendario e aggiungete il codice come in figura. In sostanza avrete detto che in corrispondenza di un cambio di data sul calendario, il testo contenuto nella label deve cambiare mostrando la data selezionata**

INSTALLARE CASSINI

Dopo averlo scaricato, è sufficiente lanciare l'eseguibile. Molto semplicemente vi ritroverete i file di cassini nella directory che avete scelto per l'installazione.
A questo punto scegliete una directory nel vostro Hard Disk che utilizzerete come contenitore dei vostri progetti Web. In sostanza sarà la root del vostro webserver. Noi abbiamo scelto c:\cassiniweb
Nella directory che contiene gli eseguibili di Cassini create un file start.bat come segue:

```
CassiniWebServer
          c:\cassiniweb\ 80 /
```

Per fare partire cassini lanciate da riga di comando il file start.bat che avete appena creato.
In pratica con questa riga avete detto che la root / di cassini corrisponde alla vostra directory fisica c:\cassiniweb e che il server gira sulla porta 80. Se avete più di un server web installato sulla vostra macchina, potete semplicemente cambiare la porta su cui gira cassini.

NOTA

COSA C'È NELL'SDK

L'SDK è disponibile dal sito ASP.NET http://www.asp.net/ Si tratta di 131 MB di files, che includono i compilatori C#, VB.NET, C++ e JScript.NET, la documentazione, un class browser e tool vari (gestore della GAC, disassembler, ecc). Per far girare le applicazioni è sufficienti installare la versione Redistributable. È poi consigliato applicare il Service Pack 1 scaricabile all'indirizzo http://www.microsoft.com/downloads/details.aspx?displaylang=it&FamilyID=ae7edef7-2cb7-4864-8623-a1038563df23

L'AUTORE

Daniele Bochicchio è il content manager di *ASPItalia.com*, community che si occupa di ASP.NET, Classic ASP e Windows Server System. Il suo lavoro è principalmente di consulenza e formazione, specie su ASP.NET, e scrive per diverse riviste e siti. È Microsoft ASP.NET MVP, un riconoscimento per il suo impegno a supporto delle community e per l'esperienza maturata negli anni. Il suo blog è all'indirizzo http://blogs.aspitalia.com/daniele/

COSA È SUCCESSO?

Molto brevemente, il Web Server ha ricevuto una richiesta per una pagina ASP.NET. Ha inoltrato questa richiesta a un processo che a sua volta l'ha passato al compilatore appropriato presente nel .NET Framework che abbiamo installato in precedenza.
Il compilatore genera una pagina intermedia in un linguaggio particolare chiamato *MSIL*. Da questo codice MSIL viene creato un'oggetto ed eseguito in memoria da un compilatore *JIT Just In Time*.
Se non avete ben chiaro come tutto abbia funzionato, concentratevi sul codice, avremo tempo per comprendere meglio i meccanismi di compilazione delle pagine ASP .NET in futuro.

LE DIFFERENZE TRA CLASSIC ASP E ASP.NET

Facciamo un paragone per aiutarci meglio nel comprendere quali siano i vantaggi di ASP .NET. Con Classic ASP è praticamente impossibile programmare secondo le regole dell'OOP *(Object Oriented Programming)*.
ASP.NET invece supporta la programmazione OO senza problemi, perché i linguaggi utilizzati (C# e VB.NET tra tutti) sono pensati secondo questa filosofia.
Classic ASP interpreta le pagine, dunque ogni richiesta al web server vuol dire molto semplicemente il parsing di tutte le righe della pagina, mentre ASP.NET utilizza una compilazione su richiesta, per cui le pagine sono compilate in automatico, attraverso un meccanismo che tiene traccia delle modifiche subite dagli oggetti dell'applicazione e provvede alla ricompilazione qualora fosse necessario.
In generale ASP risente dei problemi del mondo COM, mentre con ASP.NET abbiamo tutti i vantaggi del .NET Framework, tra cui la possibilità di far convivere più versioni dello stesso assembly ovvero, banalizzando, delle stesse classi e la possibilità di farne il deployment semplicemente copiando i file all'interno di una determinata directory.

COME FUNZIONA LA COMPILAZIONE JIT DI PAGINE ASP.NET

Uno tra i vantaggi più grandi di ASP.NET è che tutte le pagine vengono compilate al volo, alla prima richiesta o quando subentra una modifica. Il meccanismo di compilazione favorisce un incremento delle performance dell'applicazione web, perché le richieste vengono tenute, già compilate in assembly, all'interno di un percorso ben definito sul disco eseguite quindi, in MSIL, dal JITter.
Ad occhio la prima richiesta ad una pagina ASP.NET è leggermente più lenta delle successive, perché c'è un overhead aggiunto appunto dalla fase di compilazione.
Durante questo momento, infatti, viene richiamato il compilatore specifico per il linguaggio utilizzato e tutto il codice presente nella nostra pagina, sia sotto forma di codice che di HTML, viene convertito in una classe, con istanze dei vari oggetti che abbiamo utilizzato, e viene creato un *assembly* corrispondente alla pagina, che in genere viene chiamato con il nome della stessa. Fino ad una eventuale modifica (o riavvio del processo) ASP.NET continuerà ad utilizzare la versione già compilata delle pagina, con un accesso alla stessa istantaneo.
Si può notare una lentezza anche marcata in alcuni casi e questo dipende, essenzialmente, dalla potenza del processore e dalla quantità di RAM installata nel sistema.
Nel caso di un server questi due fattori in genere sono ampiamente dimensionati e non ci sono problemi, mentre su postazioni di sviluppo si potrebbe risentire una certa lentezza se le risorse hardware non sono dimensionate a dovere.
Proprio perché la compilazione è automatica, non c'è nessun vincolo particolare su un editor da utilizzare, per cui è possibile editare le proprie pagine con uno qualsiasi di quelli disponibili.

CONCLUSIONI

Questa serie dedicata ad ASP.NET proseguirà già dalle prossime puntate analizzando, con alcuni progetti da portare a termine, le caratteristiche di ASP.NET.
Nel frattempo il consiglio migliore per trovarsi preparati è quello di cominciare a fare conoscenza con l'ambiente, in modo particolare con i concetti dell'*Object Oriented Programming* e soprattutto con le sintassi e le specifiche dei due linguaggi principali, C# e VB.NET.
Ovviamente non occorre conoscerli entrambi, anche se sfruttando alla fine le stesse classi del framework, non è così difficile.

Daniele Bochicchio

Flash Action Script 2.0 Revolution

Flash e gli oggetti

Programmare ad oggetti con Flash non è più un miraggio. Action Script 2.0 catapulta la programmazione a oggetti nel meraviglioso mondo del multimedia. Vediamo come.

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste

ActionScript 1.0 ed una discreta padronanza nell'utilizzo di Flash Mx e ActionScript 1.0.

Software

Flash Mx 2004 Porfessional

Impegno

Tempo di realizzazione

Sin dall'uscita di Flash 5 gli sviluppatori professionisti e non che utilizzano questo software hanno cominciato ad avere un approccio diverso, da programmatori, durante la creazione dei loro filmati. Con ActionScript 2.0 il sogno di poter scrivere le applicazioni seguendo il paradigma della programmazione orientata agli oggetti è diventato realtà. Esploreremo insieme le novità principali della nuova release del linguaggio e affronteremo uno degli argomenti fondamentali per saper affrontare e risolvere le problematiche derivanti dalla programmazione in Flash: la tipizzazione delle variabili e i datatype primitivi.
La nuova release di ActionScript, formalmente nota come ActionScript 2.0 o indicata con la sigla AS2, è stata introdotta in entrambe le versioni di Flash Mx 2004 e, oltre ad alcune nuove funzionalità e comandi inseriti, ha segnato un cambio di direzione nella programmazione di applicazioni Flash.
Mentre in Flash Mx i concetti legati alla OOP venivano solo simulati, la nuova versione mette a disposizione alcune keyword, come ad esempio *class*, che permettono di lavorare a tutti gli effetti con classi di oggetti. Quello che manca ad AS2 per essere totalmente aderente a questa filosofia di programmazione sono alcune keyword utilizzate nella definizione delle classi, come ad esempio *protected*.

VARIABILI, TIPI DI DATI E STRICT DATATYPING

In un filmato in cui si fa uso dell'ActionScript capita spesso di voler accedere a determinate informazioni o di avere la necessità di memorizzare dati per poi utilizzarli nuovamente durante la riproduzione del file. Memorizzare i dati nelle variabili è l'unico modo per avere facilmente la possibilità di accedere e/o manipolare i dati necessari durante l'esecuzione

LA PROGRAMMAZIONE A OGGETTI

La programmazione orientata gli oggetti (O.O.P.) è un paradigma di programmazione evolutosi dal precedente paradigma della programmazione procedurale. La O.O.P. prevede di raggruppare in un'unica entità (la classe) sia i dati che le procedure che operano su di essi, creando per l'appunto un "oggetto" software dotato di proprietà (dati) e metodi (procedure) che operano sui dati dell'oggetto stesso.

DICHIARARE UNA CLASSE

```
class Test{
}
```

1 **Per definire una classe in ActionScript 2.0 si deve creare un file esterno con estensione .as con lo stesso identico nome della classe (case sensitive) all'interno del quale si utilizza la keyword class seguita dal nome della classe e dall'apertura e chiusura delle parentesi graffe. Per creare il file Test.as potete utilizzare un qualsiasi editor o l'editor integrato di Flash Mx 2004.**

Create New
Flash Document
Flash Slide Presentation
Flash Form Application
ActionScript File

delle applicazioni ed è per questo motivo che vengono definite come i principali contenitori di informazioni.
Immaginate una variabile come un contenitore vuoto all'interno del quale è possibile raccogliere qualsiasi tipo di informazione ed al quale è associato un nome che ci mette nelle condizioni di riferirci ad essa in maniera inequivocabile. Una volta che è stata creata una variabile è possibile inserirvi dati all'interno tutte le volte che vogliamo.
La caratteristica più importante delle variabili è che ci danno la possibilità di riferirci a dati e informazioni che vengono calcolati e modificati quando viene riprodotto un filmato senza però cambiare il nome utilizzato come riferimento.
Utilizzare questo riferimento fisso ci permette di fare calcoli complessi, tenere traccia delle carte utilizzate in un gioco, memorizzare i messaggi di un guestbook, influenzare lo spostamento della testina di riproduzione in base al verificarsi di alcune condizioni, ecc. ed è per questo che la conoscenza e l'uso delle variabili permette di creare applicazioni altamente interattive. Il processo con il quale viene creata una variabile si chiama dichiarazione e consiste semplicemente nel dichiarare il nome della variabile

```
var nome;
```

La keyword *var* che precede il nome della variabile, come vedremo nel dettaglio in seguito, avverte l'interprete che stiamo dichiarando una variabile con un determinato nome e che, nell'istante in cui viene creata, contiene il valore speciale *undefined*.
Questo tipo di valore indica la mancanza di dati memorizzati nella variabile, ovvero indica che ancora nessun dato è stato memorizzato al suo interno. In ActionScript è anche possibile utilizzare una tecnica definita come dichiarazione multipla di variabili

```
var nome, cognome, telefono;
```

www.flashmx2004.it
www.actionscript.it
www.risorseflash.it

IL COSTRUTTORE

```
class Test{
  function Test(){
  }
}
```

2 **Quando creiamo una nuova istanza di un oggetto generalmente lo vogliamo inizializzare assegnando dei valori di default ad alcune delle sue proprietà ed eseguendo delle operazioni preliminari; per far questo definiamo nella nostra classe una funzione chiamata costruttore. Questa funzione ha lo stesso nome della classe e viene eseguita automaticamente tutte le volte che viene creata una nuova istanza.**

DEFINIRE LE PROPRIETÀ DI UNA CLASSE

```
class Test{
private var message:String;
  function Test(){
    message = "Hello world!";
  }
}
```

3 **Le proprietà di una classe si dichiarano prima del costruttore sotto forma di variabili e possono essere dichiarate come membri public, private e static della classe. Le buone norme di programmazione orientata agli oggetti consigliano di tipizzare anche le proprietà in modo tale da essere avvantaggiati nella scrittura del codice e nel debug delle nostre applicazioni. Andiamo avanti nella stesura della classe Test e definiamo una proprietà nella quale andremo a memorizzare una stringa. L'assegnazione della stringa alla proprietà avverrà dentro il costruttore.**

DEFINIRE I METODI DI UNA CLASSE

```
public function sayHello ():String{
  return message;
}
```

4 **I metodi sono le "azioni" che un'istanza di una classe può compiere. Per definire un metodo è sufficiente dichiarare una funzione all'interno della classe definendone gli argomenti e l'eventuale data type del valore restituito dal metodo. I metodi possono essere public, private e static. Se si omette nella definizione del metodo una di queste keyword Flash lo tratterà come public e quindi accessibile dall'esterno e utilizzabile direttamente da un'istanza della classe. Definiamo il metodo sayHello() nella classe Test che deve restituire una stringa.**

UTILIZZO DI UNA CLASSE ESTERNA

```
  var myClassIstance = new Test();
label_cta.text = myClassIstance.sayHello();
```

5 **Per utilizzare una classe all'interno di un .fla è possibile salvare il file nella stessa directory contenente il file .as e utilizzare l'operatore new per memorizzare una sua istanza in una variabile. Attraverso la sintassi a punto è possibile richiamare i metodi public della classe ed ottenere, come in questo caso, il messaggio "Hello world!". Trascinate sullo stage un'istanza del component Label e assegnategli come nome dal pannello proprietà *label_cta*. Dichiariamo una nuova istanza della classe Test e usiamo il metodo sayHello() per visualizzare il messaggio al suo interno.**

ma a volte può essere poco conveniente perché in questo modo non è possibile inserire dei commenti tra una variabile e l'altra in modo da ricordare successivamente per quale motivo le avevamo dichiarate.
La maggior parte dei linguaggi di programmazione richiede la dichiarazione della variabile prima che qualsiasi tipo di dato venga memorizzato al suo interno, l'ActionScript, che ci permette di assegnare un valore ad una variabile contestualmente alla sua dichiarazione

```
var nome = "giorgio";
```

e di modificare il tipo di dato in essa memorizzato qualora, contestualmente alla dichiarazione del nome della variabile, non sia stato dichiarato anche il tipo di dato in essa contenuto.
Una variabile può assumere qualsiasi nome, basta solo osservare alcune piccole regole per una corretta dichiarazione:

- Deve essere composto solo da lettere, numeri e underscore senza, quindi, contenere spazi, punteggiatura, ecc.
- Il primo carattere del nome deve essere una lettera o un underscore
- È opportuno non utilizzare più di 255 caratteri soprattutto per ottimizzare le performance e per rendere più facile un successivo accesso ai dati contenuti nella variabile
- I nomi delle variabili sono case insensitive cioè non sono sensibili alle maiuscole, quindi sia i caratteri minuscoli che i caratteri maiuscoli vengono considerati nella stessa maniera

I TIPI DI DATI

Il suffisso *var* con ActionScript 2.0 è diventato sempre più importante. Se prima veniva utilizzato solo per dichiarare variabili locali, che vengono rimosse dalla memoria del Player dopo che è stato eseguito il blocco di codice in cui sono dichiarate, ora questo suffisso è necessario per dichiarare correttamente variabili tipizzate.
In AS2 i tipi di dati consentiti sono diventati molti di più rispetto alla precedente versione in cui ci trovavamo ad avere a che fare solo con stringhe (*String*), booleani (*Boolean*), numeri (*Number*), *Array, MovieClip, Object, null* e *undefined.* Questi vengono considerati come *DataType* primitivi visto che sono nativi del linguaggio stesso, altri tipi di dati, già presenti nella precedente versione di Flash, sono inclusi nel Player e disponibili ovunque all'interno di un filmato *(Color, Date, TextField,* ecc.). Ora qualsiasi classe che definiamo è un tipo di dato valido così come tutti i *components* visto che ad ognuno è associata una classe differente.
Una delle principali fonti di errori nella programmazione in ActionScript era proprio la caratteristica che avevano le variabili di eseguire una conversione automatica del tipo di dati in esse memorizzato. Il problema a cui si andava incontro era che, se l'interprete di ActionScript eseguiva un'operazione non corretta e convertiva ad esempio una stringa in un numero, quando si riutilizzava la variabile all'interno del filmato, alcune operazioni potevano non essere eseguite correttamente e non veniva generato nessun messaggio di errore.
Per essere agevolati nel debug e nella gestione di questo tipo di errori potete utilizzare le capacità di ActionScript 2.0 di eseguire un controllo sul tipo di dati. Per far questo è suf-

EDITOR CONSIGLIATI

SePy
http://www.sephiroth.it

SciteFlash
http://www.bomberstudios.com/sciteflash/

JEdit
http://www.jedit.org/

I VANTAGGI DELLA NUOVA VERSIONE

I vantaggi che si ottengono dall'utilizzo della nuova versione sono molti, infatti si scrive un codice più modulare, più riutilizzabile e soprattutto una si ha una capacità maggiore di debug.
Tra le novità più importanti spiccano la gestione delle eccezioni try/catch/finally, la possibilità di definire dei class path, ovvero dei percorsi, dai quali Flash può importare le nostre classi di oggetti e la possibilità di associare una classe ad un clip filmato attraverso la finestra di dialogo dedicata ai Linkage.
Gli sviluppatori più esperti non potranno fare a meno di notare anche che in AS2 il valore undefined viene convertito in NaN se utilizzato insieme ad altri numeri o nella stringa "undefined" se usato in congiunzione con altre stringhe, che una stringa non vuota, quindi diversa da "", viene sempre e comunque convertita nel valore Booleano true se usata nelle espressioni condizionali e che variabili, nome di funzioni, ecc. sono case sensitive ed ora è la variabile Pippo è quindi considerata diversa dalla variabile pippo.

ficiente dichiarare la variabile e il tipo di dato in essa contenuto

```
var nome:String = "Giorgio";
```

Dopo che l'interprete legge questa dichiarazione un tentativo successivo di memorizzare ad esempio un numero nella variabile nome genera questo messaggio di errore nella finestra di output: *Type mismatch in assignment statement: found Number where String is required.* Vi consiglio comunque di eseguire la dichiarazione del tipo di dati anche quando si dichiarano funzioni e all'interno delle classi personalizzate che creerete sia per essere più aderenti possibili alle buone norme di programmazione orientata agli oggetti, sia per farvi aiutare da Flash che, in compilazione, esegue il controllo dei tipi assegnati (type checking).

GLI ARRAY

Cominciamo ad esaminare tre DataType che in ActionScript vengono usati spessissimo: *Array, String* e *MovieClip*. Gli *Arrays* sono un tipo di dato composito e vengono utilizzati per memorizzare e manipolare liste ordinate contenenti informazioni di varia natura.

Fig. 1: ***Per avviare l'editor interno usare /file/new/actionscript***

Un *Array* può contenere qualsiasi numero di elementi anche appartenenti a tipi di dati differenti che vengono memorizzati nei così detti indici dell'*Array*. Per questo motivo possiamo definire un *Array* come una collezione sequenziale di variabili. Per fare un parallelo con la vita di tutti i giorni, immaginate di trovarvi di fronte alla cassettiera della vostra camera da letto. Pur essendo un unico "oggetto", all'interno dei cassetti sono contenuti diversi tipi di indumento… per riferirci ad un cassetto usiamo un numero ordinale, primo, secondo, terzo, ecc., per riferirci ad un elemento di un *Array* usiamo un numero cardinale uno, due, tre, ecc. ma contiamo partendo da zero: il primo cassetto è l'indice 0 dell'array a cui vogliamo accedere. Un Array può essere dichiarato utilizzando l'operatore *new* o in forma letterale utilizzando solo l'operatore di accesso all'Array, ovvero le parentesi quadre. Le due righe di codice riportate sono equivalenti

```
var person:Array = new
        Array("Giorgio", "Natili", 30);
var person:Array ["Giorgio", "Natili", 30];
```

È possibile recuperare il numero di elementi da cui è formato l'Array utilizzando la proprietà lenght ,unica proprietà della classe Array, mentre per accedere ad uno degli indici dell'Array si utilizza la sintassi

```
person[1];
```

Per assegnare un valore ad un determinato indice si usa la stessa sintassi seguita dall'operatore di assegnazione (=) e il valore che si vuole inserire.
Gli elementi di un Array possono essere anche definiti attraverso un nome anziché attraverso un indice numerico

```
var person:Array = [];
person["name"] = "Giorgio";
person["surname"] = "Natili";
person["age"] = 30;
```

In altri linguaggi di programmazione questo tipo di Array viene definito *array associativo*, ma in ActionScript questo tipo di Array viene solo simulato: un indice definito attraverso un nome non è nient'altro che una proprietà di un oggetto alla quale è possibile accedere anche utilizzando la sintassi a punto e, proprio per questo, come vedremo nei cicli di loop, si utilizza un ciclo di *for…in* per esplorare tutti gli elementi di un Array. In ActionScript gli Array possono essere ordinati sia attraverso il metodo *sort()* che attraverso il metodo *sortOn()*. Il *sortOn* esegue l'ordinamento per uno o più "campi" di un'Array associativo secondo uno o più criteri espressi attraverso una chiave numerica o attraverso una costante e separati da un OR (|). Con la parola campi indicheremo da questo momento in poi gli indici definiti attraverso una stringa

```
1    o Array.CASEINSENSITIVE
```

GLOSSARIO

VARIABILE
Una variabile è un contenitore di dati

ARRAY
Contenitore ordinato di variabili

METODO
Azioni ncompiute da un'stanza della classe ed in essa definite

PROPRIETÀ
Variabili definite in una classe di oggetti in cui si memorizzano dati di un determinato tipo

CLASSE
modello per la creazioni di oggetti (istanze) che hanno in comune tutte le proprietà e i metodi in essa definiti

COSTRUTTORE
Funzione con lo stesso nome della classe che viene richiamata al momento dell'inizializzazione dell'istanza

```
2  o Array.DESCENDING
4  o Array.UNIQUESORT
8  o Array.RETURNINDEXEDARRAY
16 o Array.NUMERIC
```

VEDIAMO UN SEMPLICE ESEMPIO

Definiamo tre Array contenti i dati di alcuni ipotetici membri di un forum e memorizziamoli in un altro Array

```
var person1:Array = [];
person1["name"] = "Giorgio";
person1["surname"] = "Natili";
person1["age"] = 30;
var person2:Array = [];
person2["name"] = "Mario";
person2["surname"] = "Rossi";
person2["age"] = 27;
var person3:Array = [];
person3["name"] = "Mario";
person3["surname"] = "Bianchi";
person3["age"] = 35;
var people:Array = [person1, person2, person3];
```

Bene, supponiamo ora di voler ordinare per nome e per cognome in ordine discendente

```
people.sortOn(["name", "surname"],
                              Array.DESCENDING);
```

Per passare più campi al metodo inseriremo questi ultimi in un Array mentre per passare un'opzione aggiungeremo il criterio separandolo con una virgola dall'Array.

LE STRINGHE

Il DataType *String* viene utilizzato per trattare e/o manipolare dati testuali (lettere, punteggiatura, ecc.) racchiusi fra virgolette. Come qualsiasi dato trattato all'interno di un computer le stringhe vengono memorizzate sotto forma di numeri e vengono poi decodificate per essere rappresentate in modo da essere comprensibili per l'utente finale. Dalla versione 6 del Player, Flash usa Unicode (*http://www.unicode.org/standard/where/*) per rappresentare le stringhe ed è per questo che è possibile gestire diverse lingue. Sulle stringhe è possibile effettuare varie operazioni sia di confronto (vedremo alcuni esempi nel paragrafo dedicato agli operatori) che di manipolazione (estrarne porzioni, verificare la presenza di un carattere, ecc.). Lavorando in maniera avanzata in Flash inizierete a trattare i *MovieClip* come veri e propri oggetti inteso come tali dal punto di vista della programmazione orientata agli oggetti, ed è quindi opportuno chiarire alcuni concetti fondamentali relativi a questa classe (dinamica). I *MovieClip* vengono implementati e trattati differentemente all'interno del Player rispetto agli altri oggetti. La differenza principale risiede nel modo in cui sono allocati e de-allocati, infatti questi, al contrario ad esempio degli Arrays, "vivono" finché sono presenti sulla timeline o fino a quando non vengono rimossi attraverso ActionScript se utilizzati in maniera dinamica (run-time). Se dichiarate un Array questo resterà presente nella memoria del Flash Player fino a quando non svuotate la variabile in cui è stato memorizzato assegnandole come valore *null*. In questo caso il Player da solo capisce che ogni riferimento deve essere rimosso: *garbage collection*. Le variabili in cui memorizziamo gli oggetti che utilizziamo nelle nostre applicazioni sono dei veri e propri puntatori a zone della memoria, le variabili in cui memorizziamo un clip filmato sono solo dei contenitori del percorso assoluto che all'interno del filmato conduce a un determinato *MovieClip*. Il differente modo di trattare queste entità è alla base del funzionamento di ActionScript, infatti, con altri tipi di oggetto, l'esistenza di un riferimento ad essi fa in modo che non vengano cancellati accidentalmente dal processo di garbage-collection interno al Flash Player. Abbiamo accennato al fatto che la classe *MovieClip* è dinamica, è proprio per questo che è possibile memorizzare variabili nei clip filmato utilizzando la sintassi a punto o richiamare funzioni memorizzate al loro interno come fossero dei metodi.

Giorgio Natili

Fig. 2: *L'editor interno di Flash è utile per generare il file delle classi*

BIBLIOGRAFIA

ActionScript the Definitve Guide For Flash Mx - Colin Moock

ActionScript 2.0 Essentials - Colin Moock

Object-Oriented Programming with ActionScript 2.0 - Jeff Tapper

I processi decisionali

Se leggo imparo

In ogni programma che si rispetti ad un certo punto ci si trova di fronte a strade diverse, le strutture decisionali consentono di stabilire quali strade prendere al verificarsi di opportune condizioni

I programmi includono spesso decisioni del tipo: se si verifica la condizione XYZ allora esegui l'istruzione *A,* Altrimenti esegui l'istruzione *B.* Questo tipo di decisione viene tradotta, in VB.Net 2003, nell'istruzione *If...Then...Else*. Se *(If)* si verifica la condizione *XYZ.* Allora *(Then)* esegui *A,* altrimenti *(Else)* esegui *B*. La condizione può essere rappresentata da una singola variabile o da un'espressione che restituisce un valore booleano *True* o *False*. In questo articolo descriveremo le strutture decisionali e gli operatori che permettono di costruire una condizione.

LE STRUTTURE DECISIONALI

Le strutture decisionali permettono di eseguire una sezione di codice, piuttosto di altre sezioni, in seguito della verifica di alcune condizioni.
In VB.Net si possono usare le seguenti strutture decisionali:

- *If...Then*
- *If...Then...Else*
- *Select Case…End Select*

L'ISTRUZIONE IF...THEN

La prima struttura è la più semplice, consente l'esecuzione di una parte di codice al verificarsi di una condizione che può assumere un valore Booleano *True* o *False*. La condizione è in genere un confronto, ma può genericamente essere costituita da una serie di espressioni legate tra loro da operatori logici. Se la condizione risulta *True*, vengono eseguite tutte le istruzioni successive alla parola chiave *Then*. Se la condizione risulta *False* l'istruzione viene ignorata.

```
If LeggoIoProgrammo Then
    ImparoVBNet()
End If
```

Se l'istruzione da eseguire è una sola, è possibile utilizzare la sintassi a riga singola senza l'istruzione *End If.*

```
If LeggoIoProgrammo Then ImparoVBNet()
```

Scriveremo un programma, in cui l'utente dovrà scrivere, in un TextBox, il numero di lenticchie che ritiene sia raffigurato in un'immagine, e successivamente dovrà cliccare con il mouse sul pulsante per conoscere l'esito del suo tentativo. Dopo aver disegnato l'interfaccia, possiamo scrivere il codice necessario al funzionamento dell'applicazione. Nella sezione delle dichiarazioni definiamo una costante che contiene il valore delle lenticchie da indovinare:

```
Const LenticchieDaIndovinare As Integer = 12128
```

Il codice necessario, alla verifica del numero di lenticchie, deve essere scritto nell'evento *Click* del pulsante *ButtonVerifica*. Questo evento, viene generato quando l'utente clicca con il mouse sul controllo. Per scrivere il codice è sufficiente fare doppio click sul controllo *Button*. Con questa operazione si aprirà la finestra del codice con il cursore posto all'interno della procedura di evento *ButtonVerifica_Click* dove scriveremo:

```
Private Sub ButtonVerifica_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                    HandlesButtonVerifica.Click
    If TextBoxNumero.Text = LenticchieDaIndovinareThen
        MessageBox.Show("Congratulazioni hai
                indovinato il numero di lenticchie")
    End If
End Sub
```

Mediante l'istruzione *If..Then* testiamo la con-

Fig. 1: La finestra del codice

dizione sul numero immesso dall'utente. Se la condizione è vera, e cioè il numero immesso dall'utente è uguale al numero di lenticchie, allora viene visualizzato il messaggio di congratulazioni. Se l'utente non indovina il numero, il programma non darà alcun riscontro. Per informare l'utente sull'esito negativo del suo tentativo, eliminandogli ogni possibile dubbio, potremo usare l'istruzione *If...Then... Else*.

L'ISTRUZIONE IF...THEN...ELSE

L'istruzione *If...Then...Else* è più completa, poiché permette di assegnare del codice alternativo, cioè del codice da eseguire se la condizione non è soddisfatta

```
If LeggoIoProgrammo Then
  ImparoVBNet()
Else
  GuardoLaTV()
End If
```

Ritornando alla nostra applicazione potremo sostituire il codice all'interno della procedura di evento *ButtonVerifica_Click* in

```
  If TextBoxNumero.Text = LenticchieDaIndovinare Then
    MessageBox.Show("Congratulazioni hai indovinato
                                il numero di lenticchie")
  Else
    MessageBox.Show("Non Hai Indovinato! Ritenta")
  End If
```

In questo modo ogni volta che l'utente scrive un numero e clicca sul pulsante, sarà visualizzato un messaggio di avviso sull'esito positivo o negativo del suo tentativo. Nel caso in cui si devono verificare più di una condizione, è possibile usare una variante con l'istruzione *ElseIf*. In questo caso, viene verificata la prima condizione, se risulta *False*, viene verificata la seconda condizione e così via fino a quando non viene individuata una condizione che risulti *True*. Una volta individuata la condizione True il compilatore eseguirà il blocco di istruzioni corrispondenti, per poi continuare ad eseguire il codice successivo all'istruzione *End If*. È possibile, inoltre, includere un blocco di istruzioni *Else*, che verrà eseguito se nessuna delle condizioni impostate risulta *True*.

```
If OggièLunedi Then
  LeggoIoProgrammo()
ElseIf OggièMercoledi Then
  LeggoUnFumetto()
ElseIf OggièVenerdi Then
  LeggoUnLibro()
Else
  GuardoLaTV()
End If
```

A questo punto possiamo migliorare la nostra applicazione, fornendo ulteriori indicazioni al concorrente sull'esito del suo tentativo:

```
Private Sub ButtonVerifica_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                        HandlesButtonVerifica.Click
  If TextBoxNumero.Text > LenticchieDaIndovinare Then
    MessageBox.Show("Il Numero di lenticchie è
                        più Piccolo di quello inserito")
    ElseIf TextBoxNumero.Text <
                        LenticchieDaIndovinare Then
      MessageBox.Show("Il Numero di lenticchie è
                        più Grande di quello inserito")
    Else
      MessageBox.Show("Congratulazioni hai
                        indovinato il numero di lenticchie")
   End If
End Sub
```

In questo caso viene confrontato il valore immesso dal concorrente con il numero di lenticchie da indovinare e viene mostrato il messaggio corrispondente. Se le istruzioni *ElseIf* crescono di numero e vengono utilizzate per confrontare la stessa espressione

INDOVINA IL NUMERO DI LENTICCHIE

Mettiamo subito in pratica i primi rudimenti creando una nuova soluzione, con la solita procedura vista nei numeri precedenti, dal nome "IndovinaQuanteSono". Possiamo ora disegnare una semplice interfaccia utente:

1 **Selezioniamo la finestra Form1 e, se non è già visualizzata, apriamo la finestra delle proprietà: Dalla finestra delle proprietà selezioniamo la proprietà Name e cambiamo subito il nome in: FormIndovina. Selezioniamo la proprietà Text e modifichiamo il testo visualizzato nella barra del titolo in: Indovina Quante sono le lenticchie raffigurate.**

NOTA

PRECEDENZE TRA OPERATORI

Nelle espressioni che includono operatori di diverso tipo, i primi ad essere valutati sono gli Operatori aritmetici in questo ordine di precedenza:

***Elevazione a potenza* (^)**
***Negazione* (–)**
***Moltiplicazione e divisione* (*, /)**
***Divisione tra interi* (\)**
***Modulo aritmetico* (Mod)**
***Addizione e sottrazione* (+, –)**
***Concatenamento di stringhe* (&)**

Successivamente vengono valutati gli operatori di confronto senza una particolare precedenza, ma in ordine da sinistra verso destra ed infine gli operatori logici nel seguente ordine:

Not
And
Or
Xor

con valori diversi, il codice potrebbe diventare illeggibile. In questi casi è consigliabile utilizzare la struttura decisionale *Select Case*.

L'ISTRUZIONE SELECT CASE

L'istruzione *Select Case* è analoga all'istruzione *If...Then...ElseIf*, ma migliora la leggibilità del codice nel caso di un numero elevato di opzioni.
L'istruzione *Select Case* si basa su un'unica condizione che viene valutata soltanto una volta all'interno del blocco. Il risultato di tale espressione viene confrontato con il valore di ciascuna istruzione *Case*, ed in caso di corrispondenza viene eseguito il blocco di codice associato.
L'esempio precedente diventa:

```
Select Case Giorno
    Case Lunedi
         LeggoIoProgrammo()
    Case Mercoledi
         LeggoUnFumetto()
    Case Venerdi
         LeggoUnLibro()
     Case Else
         GuardoLaTV()
End Select
```

In modo analogo all'istruzione *if..Then..Else If* è possibile che l'applicazione non esegua nessuna parte di codice all'interno di un'istruzione *Select Case*. Per fare in modo che l'applicazione esegua almeno una serie di istruzioni, si deve nuovamente utilizzare il comando *Else*. Nel caso dell'esempio precedente se la variabile *Giorno* è diverso da *Lunedì, Mercoledì* o *Venerdì* allora vengono eseguite le istruzioni dopo il comando *Case Else*. Utilizzando gli operatori di confronto è possibile fare in modo che l'istruzione *Select Case* controlli se una variabile rientra in un determinato intervallo. Per utilizzare gli operatori di confronto è necessario utilizzare la parola riservata *Is* (se viene omessa, Vb.Net la inserisce in automatico). Nella nostra applicazione possiamo riscrivere l'istruzione *if.. Then..Else If* in:

Fig. 2: *L'applicazione in esecuzione*

```
    Select Case CInt(TextBoxNumero.Text)
         Case Is > LenticchieDaIndovinare
              MessageBox.Show("Il Numero di lenticchie
                          è più Piccolo di quello inserito")
         Case Is < LenticchieDaIndovinare
```

2 Selezioniamo un controllo TextBox dalla casella degli strumenti (nella sezione Windows Form) e disegniamolo sulla form.

4 Selezioniamo un controllo Button dalla casella degli strumenti e disegniamolo sulla form

6 Selezioniamo un controllo PictureBox dalla casella degli strumenti e disegniamolo sulla form.

3 Selezioniamo il controllo Textbox e, dalla finestra delle proprietà, evidenziamo la proprietà Name per cambiare il nome in: TextBoxNumero.

5 Tramite la proprietà Name del controllo Button cambiamone il nome in: ButtonVerifica. e il testo in: tramite la proprietà Text in Verifica

7 Inseriamo l'immagine nel PictureBox utilizzando la proprietà Image e cliccando sul pulsante con i tre puntini che appare al lato della proprietà

Per visualizzare la finestra delle proprietà si può:

- Selezionare la voce di menu: *Visualizza /Finestra Proprietà*
- Premere il pulsante corrispondente sulla barra degli strumenti.
- Premere il tasto *F4*
- Cliccare con il tasto destro sull'oggetto e selezionare dal menu a tendina la voce *Proprietà*

La casella degli strumenti (*Toolbox*), contiene gli oggetti che è possibile aggiungere alle form in fase di progettazione. Per visualizzarla si può:

- Selezionare la voce di menu: *Visualizza /Casella degli strumenti*
- Premere l'icona *Casella* degli strumenti sulla barra degli strumenti standard
- Premere i tasti *Ctrl+Alt +X*

Per visualizzare la finestra del codice si può:

- selezionare la voce di menu: *Visualizza/Codice*
- Premere il tasto *F7*
- Cliccare sull'icona *Codice* nella finestra *Esplora soluzioni*

```
                MessageBox.Show("Il Numero di lenticchie
                                  è più Grande di quello inserito")
            Case Else
                MessageBox.Show("Congratulazioni hai
                            indovinato il numero di lenticchie")
        End Select
```

Per essere sicuri del risultato, abbiamo introdotto la funzione di conversione *Cint* per convertire il testo immesso dall'utente nel corrispondente valore di tipo Intero. È ammessa la possibilità di scrivere istruzioni *If..Then..Else* all'interno di una istruzione *Select Case* oppure un'istruzione *Select Case* all'interno di una istruzione *If..Then..Else*. Ad esempio, se nel nostro programma, vogliamo aiutare il concorrente informandolo sull'ordine di grandezza del numero da indovinare, possiamo visualizzare un messaggio di aiuto che lo informi se il numero inserito nel TextBox è due ordini di grandezza più grande o più piccolo.

```
Private Sub ButtonVerifica_Click(ByVal sender As
            System.Object, ByVal e As System.EventArgs)
                              HandlesButtonVerifica.Click
   Select Case CInt(TextBoxNumero.Text)
      Case Is > LenticchieDaIndovinare
        If CInt(TextBoxNumero.Text) > 100 *
                             LenticchieDaIndovinare Then
            MessageBox.Show("Il Numero di lenticchie è
                     Molto più Piccolo di quello inserito")
        Else
            MessageBox.Show("Il Numero di lenticchie
                          è più Piccolo di quello inserito")
        End If
            Case Is < LenticchieDaIndovinare
            If CInt(TextBoxNumero.Text) <
                        LenticchieDaIndovinare / 100 Then
               MessageBox.Show("Il Numero di lenticchie
                    è Molto più Grande di quello inserito")
            Else
               MessageBox.Show("Il Numero di lenticchie
                          è più Grande di quello inserito")
            End If
            Case Else
                MessageBox.Show("Congratulazioni hai
                            indovinato il numero di lenticchie")
      End Select
End Sub
```

Quantunque non esista un limite teorico alle istruzioni *If..Then..Else* o *Select Case* che possono essere nidificate le une nelle altre, è buona norma, per la semplicità del codice, non abusarne.

GLI OPERATORI

La condizione da testare nell'istruzione *If..Then..Else*, o *Select Case* può essere una qualsiasi espressione che restituisca un valore booleano, definita con gli operatori che VB.Net mette a disposizione. In particolare VB.Net fornisce tre tipi di operatori:

- **Aritmetici**
- **Logici**
- **Di confronto**

Gli operatori Aritmetici

Gli operatori aritmetici permettono di usare il computer come una calcolatrice, e possono essere usati per generare una condizione all'interno di un'istruzione *If..Then..Else*. VB.Net 2003 supporta i quattro operatori aritmetici fondamentali: *Addizione(+)*, *Sottrazione(-)*, *Moltiplicazione(*)*, *Divisione* (/).
Per sommare tre variabili numeriche ed assegnare il risultato dell'operazione ad un'altra variabile *Risultato*, si può ad esempio scrivere:

```
Dim Op1 As Integer
Dim Op2 As Integer
Dim Op3 As Integer
Dim Risultato As Integer
Op1 = 55
Op2 = 45
Op3 = 10
Risultato = Op1 + Op2 + Op3  'Risultato conterrà il
                                              valore 110
```

Con le prime quattro istruzioni vengono dichiarate quattro variabili (*Op1, Op2, Op3, Risultato*) di tipo intero, nelle successive tre istruzioni vengono assegnati alcuni valori alle variabili *Op1*, *Op2* e *Op3* e nell'ultima istruzione viene assegnato alla variabile *Risultato* la somma di *Op1*, *Op2* e *Op3*. Lo stesso procedimento vale per gli altri operatori. Una combinazione qualsiasi degli operatori matematici può essere usata all'interno di un'istruzione *If..Then..Else*, ad esempio.

```
If Op1 + Op2 - Op3 = 0 Then
   EseguiFunzione1()
Else
   EseguiFunzione2()
End If
```

Oltre ai quattro operatori aritmetici fondamentali, VB.Net supporta:

- l'operatore esponenziale (^) che calcola l'ennesima potenza di un numero.
- l'operatore di divisione a numero intero (\) che restituisce un risultato intero troncando la parte decimale.
- l'operatore *Mod* che restituisce il resto di una divisione tra due numeri. Spesso tale operatore viene usato per testare se un numero è multiplo

di un altro numero, poiché in questo caso il risultato è zero.

Gli operatori logici

Gli operatori logici restituiscono i valori booleani *True* e *False* e sono gli operatori più usati, insieme all'istruzione *If…Then…Else*, per prendere delle decisioni all'interno del codice. Gli operatori più comuni sono: *And , Or, Not, Xor.* Gli operatori *And, Or* e *Xor* confrontano due valori *True* o *False* e restituiscono un nuovo valore *True* o *False* calcolato in base alle tabelle riportate di seguito.

L'operatore Not

```
ImparoVbNet = Not (GuardoLaTV)
```

L'operatore *Not* modifica un valore *True* in un valore *False* e viceversa. Per maggiore chiarezza di norma si utilizzano le parentesi

L'operatore AND

```
ImparoVbNet = LeggoIoProgrammo And MiEsercito
```

La variabile *ImparoVbNet* sarà pari a *True* soltanto se sono *True* sia *LeggoIoProgrammo* sia *MiEsercito*

ImparoVbNet	LeggoIoProgrammo	MiEsercit
True	*True*	*True*
False	*True*	*False*
False	*False*	*True*
False	*False*	*False*

L'operatore OR

```
ImparoVbNet = LeggoIoProgrammo Or MiEsercito
```

La variabile *ImparoVbNet* sarà pari a *True* quando una qualsiasi delle due variabili *LeggoIoProgrammo* e *MiEsercito* sono *True*

ImparoVbNet	LeggoIoProgrammo	MiEsercit
False	*True*	*True*
True	*True*	*False*
True	*False*	*True*
False	*False*	*False*

L'operatore XOR

```
ImparoVbNet = LeggoIoProgrammo Xor MiEsercito
```

La variabile *ImparoVbNet* sarà pari a *True* quando soltanto una (e non tutte e due) delle variabili *LeggoIoProgrammo* e *MiEsercito* sono *True.*

ImparoVbNet	LeggoIoProgrammo	MiEsercit
True	*True*	*True*
True	*True*	*False*
True	*False*	*True*
False	*False*	*False*

Gli Operatori di Confronto

Gli Operatori di Confronto vengono utilizzati per confrontare operandi sia di tipo numerico sia di tipo stringa:

> *Maggiore di*
>= *Maggiore o uguale a*
< *Minore di*
<= *Minore o uguale a*
= *Uguale a*
<> *Diverso da*

Il risultato sarà un valore Booleano pari a *True* se la condizione di confronto è soddisfatta, oppure pari a *False* se la condizione di confronto non è soddisfatta, ad esempio:

```
Dim Temperatura As Integer
Dim TemperaturaMite As Integer
Dim FaMoltoCaldo As Boolean
Temperatura = 30
TemperaturaMite = 23
If Temperatura > TemperaturaMite Then
    FaMoltoCaldo = True
Else
    FaMoltoCaldo = False
End If
```

In questo caso, poiché il valore della variabile *Temperatura* è maggiore del valore della variabile *TemperaturaMite* alla fine del blocco *If..Then..Else* il valore della variabile *FaMoltoCaldo* sarà pari a *True*. Tutti gli operatori possono essere usati in combinazione tra loro (anche di tipo diverso), in questo caso il compilatore valuta e risolve ciascuna parte dell'espressione secondo un ordine prestabilito (vedi box).
Per forzare l'ordine di precedenza degli operatori e fare in modo che parti di un'espressione vengano eseguite prima di altre, è possibile utilizzare le parentesi tonde.
Gli operatori tra parentesi tonde hanno sempre la precedenza rispetto agli altri, ed in ogni caso all'interno delle parentesi tonde, vengono rispettate le regole di precedenza.

CONCLUSIONI

Ci siamo occupati delle strutture presenti in VB.Net che consentono di eseguire determinate porzioni di codice piuttosto di altre, ed abbiamo descritto gli operatori aritmetici, logici e di confronto che permettono di costruire le espressioni di controllo.
Nel prossimo articolo ci occuperemo delle altre strutture presenti in VB.Net 2003.

NOTA

Esistono due tipi di variabili, quelle definite dal programmatore e quelle definite da VB .NET come proprietà di ciascun oggetto presente sulla form. In pratica ogni volta che si crea una nuova finestra, e che viene disegnato un oggetto per comporre l'interfaccia utente, VB crea in automatico le proprietà dell'oggetto disegnato, impostandole ai valori di default.

Quando si lavora con le stringhe e gli operatori di confronto, spesso s'incontra il problema della sensibilità alle lettere maiuscole e minuscole. Nel momento in cui si confrontano delle stringhe, Vb.Net considera i caratteri maiuscoli e minuscoli, in ogni caso, come caratteri completamente diversi. Ad esempio, di fronte alla condizione: "*Abc*" = "*abc*" otterremo il risultato pari a *False.* Se in un confronto tra stringhe non è importante la differenza tra i caratteri maiuscoli e minuscoli si può utilizzare una delle tante funzioni della classe *String,* che vedremo in uno dei prossimi articoli (al posto dell'operatore di base "=").

Impara a muoverti tra le librerie standard di Java

Come API al miele

Si narra che Sansone avesse la propria forza nei capelli, e che diventò debole quando i capelli gli furono tagliati. Anche Java ha i capelli: sono le API, le librerie standard del linguaggio

Utilizza questo spazio per le tue annotazioni

Java è un ragazzo di talento: ha una sintassi interessante e tante caratteristiche utili come la Virtual Machine e il garbage collector. Ma non farebbe molta strada senza le sue librerie. Se cerchi su Internet, puoi trovare migliaia di classi precotte per cucinare i tuoi programmi Java. Ma probabilmente all'inizio non ne avrai bisogno, perché le librerie incluse nel linguaggio sono così complete che potresti scrivere intere applicazioni senza usare altro. E sono così vaste che probabilmente non le conoscerai mai tutte. Io, almeno, non l'ho mai fatto. Alcune librerie sono più importanti di altre. In questo articolo ci concentreremo soprattutto sul package *java.lang* e sulla classe *Object*. Ma prima sarà bene dare uno sguardo d'insieme alle API di Java.
Questo mese:

- Imparerai ad orientarti tra le API
- Scoprirai qualcosa di nuovo sulla classe Object
- Imparerai ad usare le classi "wrapper"

IL PRIMO APPROCCIO

Per prima cosa, apri la documentazione HTML delle API di Java 2 Standard Edition (le istruzioni per trovarla sono nelle barre laterali di questo articolo). Ti troverai davanti alla lista dei package distribuiti con J2SE. Come vedi non sono pochi. Nota che la gran parte dei package standard hanno nomi che non seguono le convenzioni di Java. L'inizio del nome del package dovrebbe sempre essere un indirizzo Internet "invertito"; ma la gran parte dei package standard hanno nomi che iniziano con il prefisso *"java"* o *"javax"*. Altri package nelle API, come *org.xml.sax*, sono stati sviluppati da terze parti e usano la nomenclatura standard. Se hai letto gli articoli dei mesi scorsi, conosci già alcuni package principali. Il più importante di tutti è il package di default *java.lang*. Contiene le classi fondamentali per il linguaggio, senza le quali Java non potrebbe nemmeno funzionare. Le classi di *java.lang* sono visibili per default da tutti i programmi Java, anche se non importi esplicitamente il package. Un altro package che forse ti suonerà familiare è *java.util*. Contiene classi di utilità generica, come *Date* (una data) o *Arrays* (una raccolta di metodi statici per gestire gli array, ad esempio per confrontarli o ordinarli). Ma *java.util* è famoso soprattutto per la classe *Collection* e le sue sottoclassi. Le collezioni di Java, come *List* e *Map*, sono definite e implementate qui dentro. È per questo che *java.util* è forse il package più comunemente importato dopo l'"obbligatorio" *java.lang*. Esistono altri package senza i quali Java non sarebbe lo stesso. Il package *java.io*, del quale parleremo a lungo il mese venturo, serve per gestire l'Input/Output (ad esempio i file). Il package *java.net* contiene le classi indispensabili per lavorare con la rete, come Socket e URL. Sono importanti anche le librerie grafiche *Swing*, contenute nel package *javax.swing* e nei suoi numerosi sottopackage. Le Swing sono il modo ufficiale per costruire interfacce GUI in Java. Il package *javax.swing* contiene classi come *Window* e *Button*, ed è talmente ricco che non ho mai incontrato nessuno che ne conoscesse tutti i dettagli. Se tutti questi package non ti bastano puoi scaricare la versione "Enterprise" di Java, nata per costruire applicazioni gestionali, che contiene librerie molto più vaste. O magari puoi scaricare qualcuna delle API specialistiche distribuite a parte da Sun, come quelle per il messaging o quelle per la grafica 3D. Ma per questo mese ti lascerò il piacere di esplorare e mi concentrerò sul nocciolo delle API: il package *java.lang*.

CARA MAMMA

La regina di *java.lang* è sicuramente *Object*, la mamma di tutte le classi Java. Abbiamo parlato di *Object* il mese scorso: è quella classe dalla quale tutte le altre ereditano, anche quando il programmatore non lo dichiara esplicitamente. Prova a leggere la pagina dedicata ad *Object* nella documentazione delle API. Come quelle di tutte le altre classi, questa pagina contiene tra l'altro un elenco di tutti i membri non privati: campi, metodi e costruttori. Nota che Object è una classe concreta, non una classe astratta.
Quindi un compilatore Java accetta questa istruzione:

```
Object o = new Object();
```

(Può sembrare un po' inutile costruire un generico Object, ma in alcuni casi particolari capita di farlo). Dicevamo che la gerarchia delle classi in Java è un albero che ha una sola classe come radice. Questa è una bella cosa, per due motivi. Il primo motivo è che questa unica radice ci permette di manipolare qualsiasi oggetto, anche se non sappiamo cos'è. L'esempio tipico sono i metodi che inseriscono e recuperano gli elementi delle collezioni. Se vai nella documentazione della classe *java.lang.LinkedList*, troverai metodi come questo:

```
class LinkedList...
  public void add(Object o);  // aggiunge un oggetto
                                              alla lista
```

Una *LinkedList* può contenere qualsiasi oggetto. Anche il valore di ritorno del metodo *LinkedList.get()*, che recupera un elemento dalla lista, è un generico *Object* (questo comporta qualche problema, come saprai se hai letto l'articolo del mese scorso). Esiste un secondo motivo per cui conviene avere una sola radice: ogni oggetto ha a disposizione i metodi definiti da *Object*. Infatti gli autori di Java hanno infilato in questa classe una serie di metodi utili o indispensabili a quasi tutti gli oggetti. La maggior parte dei metodi di Object non sono fatti tanto per essere usati così come sono, quanto per essere ridefiniti dalle sottoclassi: il fatto che siano definiti in Object significa che sono nell'interfaccia di tutti gli oggetti, ma l'implementazione di Object è poco più che un "segnaposto".
Ad esempio, il metodo *Object.equals()* confronta l'oggetto con un secondo oggetto e ritorna *true* se i due oggetti sono uguali. È una cosa che si fa spesso. Ma se sbirci nel codice sorgente delle API, distribuito con il J2SDK, scoprirai che l'implementazione di *Object.equals()* è semplicemente:

```
public class Object...
  public boolean equals(Object obj) {
    return (this == obj);
}
```

Quindi i due oggetti sono uguali solo quando sono proprio lo stesso oggetto in memoria, cioè quando sono identici. Allora a che serve questo metodo, visto che potremmo usare semplicemente l'operatore di uguaglianza? Per capirlo, guarda in che modo la classe *String* ridefinisce il metodo:

```
public class String...
  public boolean equals(Object anObject) {
    if (this == anObject) {
      return true;
    }
    if (anObject instanceof String) {
      // L'altro oggetto e' a sua volta una stringa.
      // Segue un algoritmo (che non trascrivo) che
                                              confronta
      // le due stringhe carattere per carattere e
                                       restituisce true
      // se le due sequenze di caratteri sono uguali
      <...>
    }
    return false;
  }
```

Se i due oggetti sono identici, allora *String.equals()* restituisce *true*, come *Object.equals()*. Se i due oggetti non sono identici, allora *String.equals()* restituisce *true* solo se il secondo oggetto è una stringa che contiene la stessa sequenza di caratteri del primo. Questo è quello che ci aspettiamo quando confrontiamo due stringhe, ed è anche il motivo per cui le stringhe si confrontano con il metodo *equals()*, non con l'operatore ==. Se usi == ti troverai davanti a situazioni come questa:

```
String s1 = "abc";
String s2 = "abc";
System.out.println(s1 == s2); // stampa false perche'
                             si tratta di due oggetti diversi
```

Moltissime classi ridefiniscono *equals()*, perché il concetto di "uguaglianza" cambia a seconda della classe. In molti casi è importante sapere cosa vuol dire "uguale". Ad esempio, una delle collezioni nel package *java.lang* si chiama *Set* (cioè "insieme"). I Set hanno una caratteristica comune che li distingue dalle *List:* non possono contenere più oggetti uguali. Se infili un oggetto in un *Set* contiene già un oggetto uguale, allora l'operazione di *add()* sarà ignorata. Come fa il *Set* a capire se due oggetti sono uguali? Be', chiama *equals()*. Se un oggetto non ha ridefinito

OVERLOADING E OVERRIDING

L'*overloading* e l'*overriding* sono due operazioni molto diverse, ma è possibile sbagliarsi e usare l'una al posto dell'altra. Mi è capitato di scrivere un metodo con l'intenzione di fare l'*overriding* del metodo della superclasse, ma di sbagliare l'ordine degli attributi. In questo caso Java considera i due metodi completamente diversi, e la classe finisce per avere un metodo *overloaded* in due versioni diverse. Ricordiamo che l'overloading si realizza cambiando la *firma* di un metodo, cambiando quindi l'ordine o il numero di parametri accettati, mente l'overriding si effettua modificando il comportamento di un metodo ereditato.

DOVE TROVARE LA DOCUMENTAZIONE

Vai sul sito http://java.sun.com e apri la pagina "*API Specifications*". Da qui potrai accedere alle specifiche delle API per le varie versioni di Java. Noi facciamo riferimento alla versione 1.4. La 1.5 è ancora molto nuova, e non ne abbiamo parlato in questa serie di articoli. Tieni presente che le API di Java 1.5 sono sensibilmente diverse da quelle della versione precedente.
La piattaforma Java è disponibile in due edizioni principali: Java 2 Standard Edition (in breve J2SE) e Java 2 Enterprise Edition (in breve J2EE), entrambe disponibili nella versione 1.4.x. La versione J2SE include solo le caratteristiche più importanti del linguaggio, mentre la J2EE include anche tutte le librerie per applicazioni gestionali (accesso evoluto ai database, supporto XML, applicazioni distribuite, eccetera). Le classi che abbiamo citato in questi mesi sono tutte nella Standard Edition. Insomma, le API che ti interessano sono quelle di J2SE 1.4.qualcosa. Scompattala e copiala da qualche parte – tipicamente in una directory docs nel tuo J2SDK.

equals(), vale il default: due oggetti sono uguali solo se sono identici.

Esercizio 1: *Scrivi una classe* ***Utente*** *che ha uno username e una password. Ridefinisci* ***equals()*** *in modo che due* ***Utenti*** *siano uguali se hanno lo stesso username, anche se hanno password diverse. Scrivi un programma che crea utenti con diverse combinazioni di username e password e verifica il funzionamento di* ***Utente.equals()****.*

STORIA DELL'ARTE

Quasi tutti i package più importanti delle librerie di java sono sotto-package del package *java* (*java.lang, java.util*, eccetera). Le librerie grafiche *Swing* sono invece nel package *javax.swing* e nei suoi sotto-package. Ci sono motivi storici per questo.
Il package *javax* contiene le "estensioni" di Java, quelle librerie che vengono considerate opzionali. Prima di *Swing*, i programmatori Java usavano un sistema piuttosto "rozzo" di nome *AWT* per costruire le GUI. Sun introdusse *Swing* come un'estensione, ma il nuovo sistema finì per sostituire completamente il precedente, ormai inadeguato. Sun non ha cambiato nome al package di Swing per garantire la compatibilità con il codice già scritto. Per lo stesso motivo, e perché *Swing* usa ancora pezzi di *AWT*, quest'ultimo è ancora parte di J2SE (è nel package *java.awt*).

NOTA

EREDITARIETÀ OBBLIGATORIA

Un metodo astratto non ha un'implementazione (cioè non contiene codice), quindi il compilatore non saprebbe come costruirlo. Questo è il motivo per cui se una classe ha anche un solo metodo astratto, allora la classe stessa deve essere astratta. D'altra parte, una classe astratta può anche non avere alcun metodo astratto. In questo caso il comportamento delle classe è completamente definito, ma ciononostante non puoi chiamare il costruttore della classe.

DAMMI LA TUA STRINGA

Un altro metodo di *Object* che viene ridefinito spesso dalle sottoclassi è *toString()*, che non prende parametri e restituisce una stringa che in qualche modo rappresenta l'oggetto. Per default, questa stringa è composta dal nome della classe alla quale l'oggetto appartiene, seguito da un codice identificativo. Non particolarmente utile, come puoi immaginare. Il bello inizia quando le sottoclassi ridefiniscono *toString()*. Invento un esempio:

```
/**
 * Un importo (positivo) di denaro in Euro.
 */
public class Money {
    private final int euros;  // questi campi sono "final",
                                          quindi devono essere
    private final int cents;  // definiti nel costruttore e
                                   poi non possono piu' cambiare
    public Money(int euros, int cents) {
        this.euros = euros;
        this.cents = cents;
    }
    /**
     * Somma due importi.
     */
    public Money sum(Money m) {
        int sumEuros = this.euros + m.euros;
        int sumCents = this.cents + m.cents;
        sumEuros += sumCents / 100;
```

```
        sumCents = sumCents % 100;
        return new Money(sumEuros, sumCents);
    }
    public String toString() {
        return euros + "," + cents + " _";
    }
}
```

Money ridefinisce *toString()* in modo che gli oggetti vengano convertiti in stringhe interessanti.

```
public static void main(String[] args) {
  Money m1 = new Money(12, 75);
  Money m2 = new Money(4, 50);
  Money somma = m1.sum(m2);
  System.out.println(somma);
}
```

Il programma stampa:

17,25 _

Il metodo *System.out.println()* prende qualsiasi *Object*, e chiama il *toString()* dell'Object per sapere cosa stampare. Può stampare qualsiasi oggetto, perché sarà l'oggetto stesso che gli dirà come vuole essere stampato. Ricorda che nella programmazione a oggetti la cosa più importante è assegnare le responsabilità alle classi giuste. In un linguaggio procedurale, una funzione come *System.out.println()* dovrebbe probabilmente conoscere tutti i tipi di oggetto che è possibile stampare, per decidere come stamparli. In un linguaggio object-oriented, *println()* delega all'oggetto il compito di convertirsi in stringa, e poi si occupa di stampare la stringa.
Ma cosa succede se l'oggetto da stampare è già una stringa?

```
System.out.println("123");
```

Anche in questo caso, *println()* chiama il metodo *toString()* dell'oggetto che gli ho passato, che è una String.

Esercizio 2: *Come ti aspetti che sia implementato il metodo* ***toString()*** *della classe* ***String?*** *La risposta è alla fine dell'articolo.*

(Notizie aggiuntive per i curiosi: anche la classe *System* fa parte di *java.lang*, e serve a fare cose interessanti che riguardano il sistema. *System* ha un campo *public* e *static* di nome *out*, che è un *java.io.PrintStream*. La classe *PrintStream* contiene a sua volta il metodo *println()*. Se sei confuso, non preoccuparti: parleremo degli stream il mese venturo).

Esercizio 3: *Crea un Utente (vedi esercizio 1). Stampalo. Cosa salta fuori? Prova a ridefinire* ***Utente.to-***

String() per stampare qualcosa di più interessante.

Nella documentazione di *Object* troverai molti altri metodi, alcuni dei quali, come *clone()*, meriterebbero una spiegazione dettagliata. Purtroppo abbiamo poco spazio e molte classi interessanti delle quali parlare.

TENGA, INDOSSI QUESTO

Un metodo che fa qualcosa di molto generico, come collezionare o stampare gli oggetti, può funzionare con qualsiasi *Object*. Ma in Java non tutte le variabili sono oggetti. Se il tuo codice deve girare con tutti gli oggetti ma anche con tutte le variabili primitive, sorge un problema. Non puoi dire a un metodo: "prendi un oggetto oppure un int o un char". Devi scrivere del codice speciale per trattare le variabili primitive.

System.out.println() può stampare qualsiasi oggetto, ma anche qualsiasi variabile primitiva. Come è possibile? È possibile perché il metodo bara: si è portato dietro i fratelli gemelli. Esistono diverse versioni di *println()*. Una prende un Object, le altre prendono ciascuna un tipo primitivo come *int* o *double*. Quindi il metodo è "overloaded", e il sistema decide quale versione usare a seconda del tipo del parametro. In alcuni casi neanche questa soluzione è ideale. Ad esempio, come facciamo a mettere degli *int* in un *array* insieme con degli Object? Per farlo puoi usare una delle classi *"wrapper"* contenute in *java.lang*. Grazie a queste classi puoi infilare il valore primitivo in un oggetto:

```
public class UnaListaMista {

    public static void main(String[] args)
    {
        int primoIntero = 12;
        int secondoIntero = -5;
        List l = new LinkedList();
        l.add(new Integer(primoIntero));
        l.add(new Integer(secondoIntero));
        l.add("e mettiamoci dentro anche una bella
                                                stringa...");
        for(int i = 0; i < l.size(); i++)
            System.out.println(l.get(i));
    }
}
```

La classe **Integer** è il wrapper per gli *int*. Ciascun tipo primitivo ha il suo *wrapper*. Quello di *char* si chiama *Character.* Gli altri si chiamano come il tipo primitivo corrispondente, ma con la lettera iniziale maiuscola: *Double* per *double*, *Boolean* per *boolean*, eccetera. Guarda nella documentazione del package *java.lang*. Sono tutti lì.

Il metodo **toString()** di un *Integer* che contiene il valore intero *12* restituisce naturalmente "12". Quindi il programmino qui sopra può infilare dei valori interi in una collezione insieme ad altri oggetti, e poi può stampare tutti i membri della collezione. Se vuoi tornare dall'oggetto al valore primitivo (ad esempio per fare dei calcoli aritmetici), puoi sempre farlo. Tutti i *wrapper* hanno un metodo apposta, che nel caso di *Integer* si chiama *intValue()*.

```
Integer andata = new Integer(42);
int ritorno = andata.intValue();
```

CODICE CHE SI DOCUMENTA DA SOLO

La documentazione delle API di Java è prodotta con un programmino che si chiama *Javadoc*.
Lo puoi trovare nella directory bin del J2SDK. Javadoc legge i commenti al sorgente, e quando trova un commento scritto con una sintassi speciale lo trasforma in documentazione.
Per avere un esempio, prova ad aprire uno qualsiasi dei sorgenti di Java.
Ecco un frammento di codice dal sorgente di *java.util.LinkedList*:

```
/**
 * Returns the first element in this list.
 *
 * @return the first element in this list.
 * @throws
NoSuchElementException if this list is
                                    empty.
 */
public Object getFirst()
{
  if (size==0)
    throw new NoSuchElementException();
  return header.next.element;
}
```

La parte che inizia con una barra e un doppio asterisco è un normale commento per il compilatore Java, ma è anche riconosciuto come documentazione da Javadoc.
I tag come *@return* descrivono i parametri, il valore di ritorno e le eventuali eccezioni.
***Javadoc* legge i sorgenti e organizza queste informazioni in una serie di pagine HTML. In questo modo la documentazione è vicina al codice, ed è più difficile che "resti indietro" quando qualcuno decide di modificare il programma.**
Vale la pena di imparare ad usare *Javadoc*.
È uno standard utile, ma non difficile.

SI CHIUDE...

Mi resta solo da darti la soluzione all'**Esercizio 3**. Ecco il metodo *toString()* della classe *String*, direttamente dal sorgente di *Sun:*

```
class String...
    public String toString()
  {
        return this;
  }
```

Il mese venturo concluderemo questa serie introduttiva a Java con un argomento fondamentale: imparerai come accedere al file system.

Paolo Perrotta

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

CATTURARE L'OUTPUT DEI PROGRAMMI

File sul cdrom: codice/pipes.zip

Salve a tutti, questo è una dimostrazione di come catturare l'output dei programmi e mostrarli in una *TextBox*

Tip fornito dal sig. Diego Marco

```
Private Type PROCESS_INFORMATION
  hProcess As Long
  hThread As Long
  dwProcessId As Long
  dwThreadId As Long
End Type
Private Type STARTUPINFO
...
Private Declare Function CreatePipe Lib "kernel32" (phReadPipe
                    As Long, phWritePipe As Long, lpPipeAttributes As
                SECURITY_ATTRIBUTES, ByVal nSize As Long) As Long
Private Declare Sub GetStartupInfo Lib "kernel32" Alias
                  "GetStartupInfoA" (lpStartupInfo As STARTUPINFO)
Private Declare Function CreateProcess Lib "kernel32"
   Alias "CreateProcessA" (ByVal lpApplicationName As String, ByVal
            lpCommandLine As String, lpProcessAttributes As Any,
   lpThreadAttributes As Any, ByVal bInheritHandles As Long, ByVal
          dwCreationFlags As Long, lpEnvironment As Any, ByVal
      lpCurrentDriectory As String, lpStartupInfo As STARTUPINFO,
        lpProcessInformation As PROCESS_INFORMATION) As Long
Private Declare Function ReadFile Lib "kernel32" (ByVal hFile As
      Long, lpBuffer As Any, ByVal nNumberOfBytesToRead As Long,
      lpNumberOfBytesRead As Long, lpOverlapped As Any) As Long
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject
                                              As Long) As Long
Private Const STARTF_USESHOWWINDOW = &H1
Private Const STARTF_USESTDHANDLES = &H100
Private Const SW_HIDE = 0
Private Const EM_SETSEL = &HB1
Private Const EM_REPLACESEL = &HC2
Sub Excecute(ByVal cmdLine As String)
...
  cp_sa.nLength = Len(cp_sa)
  cp_sa.lpSecurityDescriptor = 0
  cp_sa.bInheritHandle = True
  pr_sa.nLength = Len(pr_sa)
  tr_sa.nLength = Len(tr_sa)
  If CreatePipe(hRead, hWrite, cp_sa, 0) <> 0 Then
    sinf.cb = Len(sinf)
    GetStartupInfo sinf
    sinf.hStdOutput = hWrite
    sinf.hStdError = hWrite
    sinf.dwFlags = STARTF_USESHOWWINDOW Or
                                          STARTF_USESTDHANDLES
    sinf.wShowWindow = SW_HIDE
    If CreateProcess(vbNullString, cmdLine, pr_sa, tr_sa, True, 0,
                          Null, vbNullString, sinf, pi) <> 0 Then
      m_out = ""
      Do
        Erase lpBuffer()
        If ReadFile(hRead, lpBuffer(0), 1023, bRead, ByVal 0&) Then
          str = Replace(StrConv(lpBuffer, vbUnicode), Chr$(0), " ")
          m_out = m_out & Trim$(str)
        Else
          Exit Do
        End If
        CloseHandle hWrite
      Loop
      CloseHandle hRead
    End If
  End If
  Text1 = m_out
End Sub
Private Sub Form_Load()
Excecute "cmd /c dir c:\*.*"
End Sub
```

SCARICARE UNA PAGINA WEB

Gentile Redazione ioProgrammo, sono un Vostro affezionato lettore, ritengo che la Vostra rivista sia tra le migliori oggi presenti sul mercato italiano nel settore IT per i contenuti e per la chiarezza e facilità con cui tratta gli articoli. Vi invio in allegato una soluzione in C# relativa ad un piccolo problema di programmazione che spesso ci affligge ovvero come poter scaricare una pagine Web da Internet. Nell'esempio di seguito riportato si utilizzano le classi *WebRequest* e *WebResponse* per una richiesta di dati da Internet e per leggere la risposta. Nello specifico si richiederà un particolare URI, ad esempio una pagina Web su un Server. La sequenza di

byte restituita verrà codificata secondo lo standard UTF-8 (UCS Transformation Format, form a 8 bit) il risutato, infine, verrà mostrata sulla console.

Tip fornito dal sig. Domenico Monteleone

```
using System;
using System.Net;
using System.IO;
using System.Text;
public class GetHtmlPage
{
  public static void Main()
  {
    Uri pageUri = new Uri("http://www.miosito.it/index.htm");
    WebRequest pageWebRequest= WebRequest.Create(pageUri);
    WebResponse pageWebResponse= pageWebRequest.GetResponse();
    Stream pageStream = pageWebResponse.GetResponseStream();
    Encoding paceEncoding = Encoding.GetEncoding("utf-8");
    StreamReader pageReader = new StreamReader(pageStream,
                                                    paceEncoding);
    string result = pageReader.ReadToEnd();
    Console.WriteLine(result);
    pageWebResponse.Close();
    pageStream.Close();
    pageReader.Close();
  }
}
```

CREARE UNA DIRETORY VIRTUALE

SUL CD ROM: codice/CVirtualDir.zip

Salve, vi invio una routine in C# per creare/aggiornare una directory virtuale su un server IIS

```
CreaVirtualDir(string strHost, string strVirtualDir, string strRealDir)
```

dove strHost indica il server su cui fare l'aggiornamento, strVirtualDir indica il nome della nuova cartella virtuale mentre strRealDir indica la cartella in cui effettivamente risiedono i file. Prima di utilizzare la funzione è necessario impostare un riferimento a *System.DirectoryServices.dll*.

Saluti.

Tip fornito dal sig. Roberto Roncato

JMENUBAR AUTOMATICHE

File sul cdrom: codice/jemenu.zip

Consiste in due semplici classi java con le quali è possibile creare una *JMenuBar* e popolarla a piacimento con altrettanti *JMenu* o *JMenuItem* semplicemente editando un file xml è quindi consigliata a chi crea appicazioni grafiche che utilizzano swing. Cuore del funzionamento è proprio il file XML che deve essere creato rispettando i criteri come si puo comprendere dall'esempio in allegato in pratica gli elementi <item> primari rappresentano i menu che si inseriscono sulla menuBar all'interno dei quali è possibile annidare altri elementi *<item>* ognuno dei quali rappresentera un altro *JMenu* oppure un *JMenuItem* a seconda dello schema di menu che vogliamo ottenere. L'accortezza più importante e' che ogni elemento *<item>* deve contenere un elemento *<name>* che darà appunto il nome al *JMenu*.

Perchè il tip funzioni è necessario procurarsi le librerie JDOM, liberamente scaricabili via internet, che consentono una semplice gestione dei file XML. Le classi che vi propongo si chiamano *MenuFromXml* e *CreaMenu* e lavorano in coppia in piu la classe *TestMenu* e il file *menu.xml* servono per provare il funzionamento. Con L'occasione vi porgo Distinti saluti e vi faccio i miei complimenti per la rivista che seguo fedelmente ormai da anni.

Tip fornito dal sig. Baldi Daniele

JAVA

COME ALLOCARE DINAMICAMENTE UN ARRAY

Java non consente di estendere la lunghezza di un array a runtime. Per ottenere una funzionalità imile bisogna ricorre ai *Vector*. Se si vogliono effettuare numerosi accessi può essere utile convertire il vettore in un array, cosi come nell'esempio seguente:

```
int count = mioVector.size();
String[] mioArray = new String[count];
mioVector.copyInto(mioArray);
```

È utile ricordare che, se il Vector non è più utilizzato, il garbage collector si occuperà senz'altro di eliminarlo automaticamente.

L'ESSENZIALE PER INVIARE UNA MAIL

Di seguito trovate il più semplice codice possibile per inviare una mail.

```
import java.io.*;
import javax.mail.*;
import javax.mail.internet.*;
import javax.activation.*;
public class SendApp
{
  public static void send(String smtpHost, int smtpPort, String from,
                          String to, String subject, String content)
  throws AddressException, MessagingException {
    // Create a mail session
    java.util.Properties props = new java.util.Properties();
    props.put("mail.smtp.host", smtpHost);
    props.put("mail.smtp.port", ""+smtpPort);
    Session session = Session.getDefaultInstance(props, null);
    // Construct the message
    Message msg = new MimeMessage(session);
    msg.setFrom(new InternetAddress(from));
    msg.setRecipient(Message.RecipientType.TO, new InternetAddress(to));
    msg.setSubject(subject);
    msg.setText(content);
    // Send the message
    Transport.send(msg); }
  public static void main(String[] args) throws Exception
  {
```

```
    // Manda un messaggio di test
    send("hostname", 25, "mario@ioprogrammo.it", "luca@edmaster.it",
                           "re: ciao", "Da quanto tempo! Come stai?"); }
  }
```

COME SAPERE SE UN THREAD HA CONCLUSO IL SUO LAVORO

Supponiamo di aver fatto partire un thread di voler aspettare fino a che il thread non abbia finito il suo compito: una possibilità è di testare continuamente il thread con il metodo isAlive(). Questo sistema funziona ma obbliga il processore a sprecare preziosi cicli di clock. Un metodo più efficiente (e più elegante!) consiste nell'utilizzare il metodo join() così come potete vedere di seguito:

```
MyThread myThread = new MyThread();
myThread.start();
try
{ myThread.join();}
catch(InterruptedException ie)
{}
System.out.println("Ho finito");
```

CREARE E RECUPERARE UN DATASET

Una funzione che accetta in ingresso i parametri sulla connessione, la selezione SQL da effettuare e ed il nome della tabella da interrogare. In uscita avremo un DataSet con riempito con il risultato dell'interrogazione. Leggendo queste poche righe, sarete in grado di connettervi a qualsiasi DB senza problemi.

```
public System.Data.DataSet dsReturnedDataSetFrom(
                              System.Data.SqlClient.SqlConnection
sqlOpenConn, System.Data.SqlClient.SqlCommand
                    TheSQLCommand, string TheQueriedTable, string
                                                   TheDatasetName)
{ TheSQLCommand.Connection = sqlOpenConn;
   System.Data.SqlClient.SqlDataAdapter NewDA = new
            System.Data.SqlClient.SqlDataAdapter (TheSQLCommand);
   DataSet NewDS = new DataSet();
   TheDatasetName.Trim();
   NewDS.DataSetName=TheDatasetName;
NewDA.Fill(NewDS, TheQueriedTable);
NewDA.Dispose();
return NewDS; }
```

IL TIP DEL MESE

C'È LA CHIAVETTA?

Vi interesserebbe conoscere se è inserita la cosiddetta "chiavetta USB"? Si possono utilizzare le classi di System.Management per monitorare l'inserimento di questi dispositivi.
Di seguito trovate una piccola applicazione console che, all'atto dell'inserimento di una chiavetta, mostra tutte le proprietà del sistema:

```
using System;
using System.Management;
class WMIEvent
{
  public static void Main()
  {
     WMIEvent we = new WMIEvent();
     ManagementEventWatcher w = null;
     WqlEventQuery q;
     ManagementOperationObserver observer =
                              new ManagementOperationObserver();
     // Bind to local machine
     ManagementScope scope =
                           new ManagementScope("root\\CIMV2");
     scope.Options.EnablePrivileges = true; //set required privilege
   try
   {
      q = new WqlEventQuery();
      q.EventClassName = "__InstanceOperationEvent";
      q.WithinInterval = new TimeSpan(0,0,3);
      q.Condition = @"TargetInstance ISA 'Win32_DiskDrive' ";
      w = new ManagementEventWatcher(scope, q);
      w.EventArrived += new EventArrivedEventHandler(
                                           we.DiskEventArrived);
      w.Start();
      Console.ReadLine(); // block main thread for test purposes
   }
   catch(Exception e)
   { Console.WriteLine(e.Message);
   }
   finally
   { w.Stop(); }
}
public void DiskEventArrived(
                          object sender, EventArrivedEventArgs e)
{
   //Get the Event object and display its properties (all)
   foreach(PropertyData pd in e.NewEvent.Properties)
   {
     ManagementBaseObject mbo = null;
     if(( mbo = pd.Value as ManagementBaseObject) != null)
     {
       Console.WriteLine("--------------Properties-----------------");
       foreach(PropertyData prop in mbo.Properties)
       Console.WriteLine("{0} - {1}", prop.Name, prop.Value);
     }
   }
  }
}
```

COME SOMMARE UN VALORE ALLA DATA CORRENTE

Le classi per la gestione delle date e del tempo possono apparire un po' complesse da utilizzare. Il codice che segue illustra come effettuare il parsing di una stringa di tempo e addizionarne il valore alla data corrente.

```
import java.util.Date;
import java.util.TimeZone;
import java.util.Calendar;
import java.util.GregorianCaength(); }
} // end class TestDateTime1
```

IL PC FA BEEP!

Prima o poi giunge la necessità di far suonare il famigerato beep: che vogliate segnalare una qualche anomalia o semplicemente richiamare l'attenzione dell'utente, quale miglior modo! In C# si fa così:

```
Console.WriteLine( "\a" );
```

COME FORMATTARE UN IMPORTO

Il codice che segue consentesi formattare la stringa di una textbox, seguendo le indicazioni di SQL Server, indipendentemente dalle impostazioni dell'utente sull'utilizzo dei decimali.

```
Public Function FormattaValuta (ByVal strValore As String) As String
    Dim DecS As String = Mid$(CStr(1.1), 2, 1)
    Dim intDot As Integer = InStr(strValore, ".")
    Dim intComma As Integer = InStr(strValore, ",")
    Select Case DecS
        Case ","
            If intDot > intComma Then
                strValore = Replace(strValore, ",", "")
                strValore = Replace(strValore, ".", ",")
            End If
            strValore = Decimal.Round(strValore, 2)
            strValore = Replace(strValore, ",", ".")
        Case "."
            If intComma > intDot Then
                strValore = Replace(strValore, ".", "")
                strValore = Replace(strValore, ",", ".")
            End If
            strValore = Decimal.Round(strValore, 2)
    End Select
    Return strValore
End Function
```

QUANTI BYTE IN UNA DIRECTORY?

Questa funzione ritorna il numero di byte utilizzati in una qualsiasi directory presente su disco:

```
Function DirUsedBytes(ByVal dirName As String) As Long
Dim FileName As String
Dim FileSize As Currency

' aggiunge un backslash se non è già presente
If Right$(dirName, 1) <> "\" Then dirName = dirName & "\"
Endif
FileSize = 0
FileName = Dir$(dirName & "*.*")
Do While FileName <> ""
    FileSize = FileSize + _
        FileLen(dirName & FileName)
    FileName = Dir$
Loop
DirUsedBytes = FileSize

End Function
```

Ed ecco come richiamare la funzione:

```
MsgBox DirUsedBytes("C:\Windows")
```

Questo mese in palio una **PENNA USB EUTRON** per archiviare software e dati

Inviaci la tua soluzione ad un problema di programmazione, una faq, un tip... Tra tutti quelli giunti mensilmente in redazione, saranno pubblicati i più meritevoli e, fra questi, scelto il Tip del mese

PREMIATO CON UN FANTASTICO OMAGGIO!

Invia i tuoi lavori a ioprogrammo@edmaster.it

Come rendere una form trasparente

Il framework .NET offre allo sviluppatore una gran quantità di classi e possibilità che spesso non vengono sfruttate fino in fondo. Anche le classi che maneggiamo quotidianamente nascondono "segrete" proprietà e metodi a volte stupefacenti. Questa volta siamo andati a scovare la proprietà di Opacità di cui godono tutte le form in .NET. Questa proprietà determina il grado ti "trasparenza" del form. A seconda del valore assunto, la form e tutti gli elementi in essa contenuti diventano più o meno trasparenti, lasciando intravedere ciò che si trova al di sotto della form. La proprietà accetta un valore *double* compreso fra *0* e *1*. A *0* corrisponde l'assoluta trasparenza. In pratica il valore *0* rende completamente invisibile la form. Man mano che il valore aumenta da *0* verso *1*, la form diventa sempre meno trasparente, fino a raggiungere il grado di totale opacità con il valore impostato a *1*, che è poi il valore di default per tutte le form. Attraverso Visual Studio .NET, è possibile accedere al valore di opacity di una form direttamente dal pannello delle proprietà, in questo caso il valore va epresso in percentuale (0% = invisibile, 100% = opaco). La proprietà può essere utile in diverse circostanze, ce ne vengono in mente due: un indicatore sempre attivo che non dia troppo fastidio, e un effetto speciale per stupire i clienti all'avvio dell'applicazione!

Raffaele del Monaco

‹1› UN NUOVO PROGETTO

In Visual Studio .NET Apriamo un nuovo progetto C# e definiamolo come applicazione Windows.
Dovremo come sempre specificare il nome del progetto ed il path completo in cui tutti i file del progetto saranno salvati.
Abbiamo scelto C# come inguaggio, ma il porting verso VB.NET è pressoché immediato.

‹2› DEFINIAMO UNA TRACKBAR

Trasciniamo una TrackBar dalla Casella degli strumenti alla nostra form, consentirà all'utente di variare l'opacità della form in modo molto diretto, garantendo un feedback immediato sul risultato.
Trasciniamo anche due label per segnare l'inizio e la fine della barra: modifichiamone la proprietà text con "Invisibile" e "Opaco".

‹3› AGGANCIAMO LA TRACKBAR IN BASSO

Selezioniamo la TrackBar sulla form e, nelle proprietà, andiamo a impostare l'ancoraggio in basso. In questo modo, anche ridimensionando la form, la nostra TrackBar occuperà sempre la parte bassa della form e la dimensione varierà in modo da occupare sempre tutta la lunghezza del form.

‹4› UN PO' DI ESTETICA

Sempre con la TrackBar selezionata, andiamone a definire alcune proprietà seguendo i valori che potete leggere in figura. Stiamo definendo sia caratteristiche "estetiche" (il numero e la distanza dei trattini lungo la barra) sia caratteristiche funzionali, cioè i valori di fondo scala.

‹5› MODIFICHIAMO LA TRASPARENZA

```
private void trackBar1_Scroll(object sender,
                              System.EventArgs e)
{
    this.Opacity = ((double)trackBar1.Value)/100;
}
```

È tempo di scrivere il nostro codice. Con un doppio click sulla TrackBar, ci troveremo nella sezione del codice dedicata alla gestione dell'evento Scroll: ogni volta che l'utente muoverà il cursore della barra, la forma reagirà cambiando istantaneamente il valore di opacità in base al valore restituito dalla barra. Da notare che effettuiamo un cast esplicito verso il tipo double in conformità con la definizione della proprietà Opacity.

‹6› L'EFFETTO FINALE

Ecco il risultato dei nostri sforzi. Provate a muovere il cursore della TrackBar verso destra e verso sinistra: il form tenderà a scomparire o a farsi più opaco di conseguenza. Provate anche a ridimensionare i margini della form: apprezzerete le proprietà dell'ancoraggio che abbiamo impostato per la TrackBar e che ne determinano la sempre perfetta utilizzabilità.

Costruiamo una libreria di classi C# con gli strumenti visuali di Visual Studio .NET

In progetti di medie e grandi dimensioni è indispensabile una corretta divisione in librerie di classi. Benché sia, in linea teorica, possibile sviluppare tutta l'applicazione in un unico pacchetto, ragioni di manutenibilità ed efficienza spingono verso un frazionamento delle classi in librerie omogenee. Per un nostro progetto potremo infatti prevedere una libreria per le funzioni matematiche, una per la gestione dell'output, un'altra ancora per la comunicazione con sistemi remoti. Questa divisione consentirà una migliore leggibilità del codice che si sarebbe altrimenti espresso come un diluvio di istruzioni in un blocco unico e di difficile interpretazione.
Anche i futuri aggiornamenti del software risulteranno agevolati dalla divisione in librerie: potremo, di volta in volta, occuparci dell'aggiornamento di porzioni specifiche del codice, con la garanzia di lasciare integre (e dunque perfettamente funzionanti) le porzioni non interessate dall'aggiornamento. Senza contare che, se saremo stati così bravi da rendere le librerie sufficientemente indipendenti, potremo più facilmente importare quel codice in successivi progetti, senza dover reinventare la ruota ogni volta… insomma: l'obiettivo primo della programmazione a oggetti. In questa occasione conosceremo gli strumenti di Visual Studio .NET che automatizzano il processo della generazione delle librerie.

Raffaele del Monaco

‹1› CREIAMO IL PROGETTO

All'avvio di Visual Studio andiamo a indicare la creazione di un nuovo progetto, selezionando Progetti di Visual C#, e indichiamo l'icona Libreria di classi. Indichiamo nome e percorso del nostro progetto. Un click su OK e siamo pronti a realizzare la nostra prima libreria in C#.

‹2› EDITIAMO LA CLASSE

All'interno di Visual Studio, in Esplora soluzioni, possiamo notare che è gia disponibile una classe di nome Class1. Un doppio click su questa classe e potremo vedere, nella finestra di editing, il codice generato automaticamente dall'ambiente.
Come per le Windows Form, già a questo punto possiamo provare a compilare la classe. Dal menu Genera, selezioniamo Genera Souluzione. Nella sotto-directory \bin\Debug, potrete osservare il file Matematica .dll appena generato.

‹3› IL PRIMO METODO

La classe è perfettamente funzionante ma… assolutamente inerte! È giunto il momento di farle fare qualcosa che, per una classe, equivale a possedere dei metodi. Dalla finestra Visualizzazione di classi, espandiamo l'albero gerarchico del progetto fino alla classe Class1. Un click con il tasto destro per selezionare la voce Aggiung->/Aggiungi metodo…

‹4› DEFINIAMO I PARAMETRI

Nel Wizard che si apre, indichiamo tutti i dettagli della firma del nostro metodo: numero e tipo di parametri accettati, senza dimenticare ovviamente il nome del metodo ed il tipo restituito dallo stesso. Nel nostro caso, il nostro semplice metodo di test accetterà due *addendi* e ne restituirà la somma. Un Click su *Fine* per concludere l'operazione.

‹5› UN RAPIDO CONTROLLO

Nel visualizzatore di classi potete ora notare che alla nostra classe è stato aggiunto un metodo che riporta fedelmente le indicazioni da noi fornite al Wizard.
Un doppio click sul metodo somma e potremo ispezionarne il contenuto come spiegato nel passo successivo.

‹6› ED ECCO IL CODICE

In figura vedete due porzioni di codice. La prima è quella generata autonomamente dal Wizard. Noterete come, oltre alla definizione dei parametri coerente con le nostre indicazioni, l'ambiente ha anche generato una istruzione di return, per rendere il metodo conforme alle specifiche C#. Noi sostituiremo proprio questa riga di codice, nel modo illustrato nella seconda porzione. Generiamo il progetto come al passo 2 e la nostra libreria è pronta per essere utilizzata.

Utilizzare una libreria di classi in Visual Studio .NET

Per imparare a utilizzare librerie di classi esterne al nostro progetto, approfitteremo della libreria appena creata nel tutorial precedente. Il progetto che svilupperemo nei sei passi canonici non realizza funzioni eccezionali: costruiremo una semplice che accetta in input due interi e ne fornisce la somma in uscita. Niente di più banale, si dirà! Eppure con questo piccolo esercizio faremo pratica con l'utilizzo di classi esterne e conosceremo un indispensabile metodo che il framework ci regala per la conversione da stringa a numero intero. Conversione da maneggiare con cura: come vedremo, essendo un'operazione ad alto rischio di errore, dovrà essere eseguita all'interno di un blocco try/catch, pronti a gestire le eventuali eccezioni sollevate a runtime.
Sarà anche l'occasione per apprezzare la comodità dell'intellisense che, come vedremo, funziona perfettamente anche con le classi personalizzate create dallo sviluppatore.
Per poter seguire i vari passi del tutorial è necessario aver realizzato il progetto descritto nel tutorial precedente.

Raffaele del Monaco

‹1› CREIAMO IL PROGETTO

Creiamo un nuovo progetto C# scegliendo la voce Dal menu File->Nuovo->Progetto…
Scegliamo di realizzare una nuova Applicazione per Windows in C# e indichiamo nome e percorso del nostro progetto.
Un click su OK e saremo pronti per cominciare.

‹2› DISEGNAMO LA FORM

Trasciniamo sulla form un pulsante e tre textBox. Posizioniamoli seguendo lo schema che potete notare in figura.
I primi due saranno le caselle di input della nostra rudimentale calcolatrice, mentre nel terzo textBox ci finirà il risultato dell'addizione.
Lasciamo invariati i nomi dei controlli, ma facciamo un po' di pulizia, cancellando il testo predefinito nelle proprietà dei textBox e cambiando il testo del pulsante.
Possiamo modificare l'etichetta del bottone usando la Property Label.

‹3› AGGIUNGIAMO IL RIFERIMENTO

È il momento di collegarci alla libreria Matematica realizzata precedentemente. Selezioniamo la voce Aggiungi riferimento… dal menu Progetto. Nella finestra che appare, attraverso il pulsante Sfoglia, potremo indicare l'esatto percorso della DLL contenente la libreria di classi. Un click su OK e ci ritroveremo le classi di Matematica fra quelle disponibili e visualizzate in Esplora soluzioni.

‹4› GESTIAMO GLI ELEMENTI

```
private void button1_Click(object sender, System.EventArgs e)
{
    int Add1=0;
    int Add2=0;
    int risultato;
    bool valido = true;

    Matematica.Class1 miaCalcolatrice = new Matematica.Class1();

    try {
        Add1 = Int32.Parse(this.textBox1.Text);
        Add2 = Int32.Parse(this.textBox2.Text);
    } catch(Exception) {
        valido = false;
        MessageBox.Show("Accettiamo solo valori interi!");
    }
```

Un doppio click sull'unico pulsante della nostra form e potremo gestire l'evento di click sul pulsante stesso. In figura vedete la prima parte di questa porzione di codice. Da notare sono: la creazione dell'oggetto miaCalcolatrice, che come vedete è un'istanza di class1, appartenente alla libreria Matematica; l'utilizzo di Int32.Parse che si occupa di trasformare in int un oggetto string. Come vedete, abbiamo chiuso tutto in un try/catch che verifica l'effettiva immissione di valori interi nei due textBox.

‹5› L'INTELLISENSE

```
if (valido) {
risultato = miaCalcolatrice.somma(
                    int Class1.somma (int addendo1, int addendo2)
```

```
    if (valido) {
    risultato = miaCalcolatrice.somma(Add1, Add2);
    this.textBox3.Text =  risultato.ToString();}
}
```

La prima immagine illustra l'intellisense all'opera: come vedete la nostra libreria è stata correttamente riconosciuta e, utilizzando i suoi metodi, siamo aiutati nella scrittura del codice attraverso l'indicazione del numero e del tipo di parametri accettati.
La seconda immagine, illustra il completamento del codice iniziato al passo 5. Poco da spiegare se non il test fatto sulla condizione di valido per evitare di passare dati inconsistenti all'oggetto *miaCalcolatrice*.

‹6› L'EFFETTO FINALE

Voila! La nostra applicazione all'opera.
Ovviamente la funzione che implementa non è nulla di particolarmente eccezionale ma, in appena due tutorial, abbiamo imparato nozioni fondamentale sulla creazione e l'utilizzo delle librerie di classi.
Queste poche righe di codice possono rappresentare la base anche di progetti di grandi dimensioni.

Query di accodamento per azzerare campi contatore

Nel comune uso dei database spesso ci ritroviamo a maneggiare campi contatore. Si tratta sicuramente di un tipo di attributo molto comodo, visto che ad esso sono legate utili funzioni. Il campo contatore, infatti, si presta ottimamente ad essere associato alla chiave della tabella poiché il continuo incremento di un'unità lo rende un numero univoco (tutti i valori sono differenti tra loro). Inoltre, non richiede un inserimento specifico poiché è lo stesso DBMS Access che si occupa dell'automatico incremento, anche se si usano le maschere di input. Sembrerebbero, quindi, solo rose e fiori! Purtroppo non è così. Qualora dalla tabella si eliminino dei record (anche tutti) eventuali nuovi input assoceranno al campo contatore valori incrementali rispetto anche alle tuple eliminate. In altri termini se prima della cancellazione dei record il contatore segnava ad esempio il numero 150; il nuovo primo record partirà con il valore del contatore pari a 151. Spesso, invece, a seguito di una cancellazione desidereremmo che il campo contatore ripartisse da 1 o comunque da un numero a nostra scelta. Questo desiderio si attua in modo un po' macchinoso mediante le query di accodamento. Si tratta di un procedimento indispensabile che il gestore di database deve conoscere.

Fabio Grimaldi

‹1› SUPPONIAMO DI AVERE UNA TABELLA LIBRO

Il problema si innesca quando un record in una tabella, che contiene dei dati viene eliminato totalmente o parzialmente. Nell'esempio dopo aver evidenziato tutte le righe, con il tasto destro del mouse vengono eliminate.
A questo punto se si prova a inserire un nuovo dato ad esso sarà automaticamente associato al campo IDL il valore 4 e non il valore 1 come ci si aspetterebbe. Provare per credere.

‹2› CREAZIONE DI UNA TABELLA DI SUPPORTO

Si inizia creando una tabella di supporto con un unico campo di tipo intero lungo (dello stesso tipo e con lo stesso nome del campo contatore di Libro).
Aprendo la tabella si inserisce il numero, nuovo punto di partenza del omonimo campo della tabella iniziale Libro. Si chiude la tabella assegnando ad essa un qualsiasi nome.

‹3› CREAZIONE DELLA STRUTTURA DELLA QUERY DI ACCODAMENTO

Dal raccoglitore degli strumenti si procede creando in modalità struttura una query.
Si aggiungono entrambe le tabelle, ossia Libro e quella di supporto (è possibile a rigore anche con la sola presenza della tabella di supporto).
Si aggiunge il campo della tabella di supporto e i restanti della tabella Libro.
Si sceglie la query di accodamento.

‹4› SALVATAGGIO DELLA NUOVA QUERY PER L'AGGIUNTA DEI DATI

Nel chiudere la query viene chiesto il nome. È opportuno assegnare lo stesso nome della tabella originale, ossia Libro.
Si seleziona il database aperto (l'altra scelta si usa qualora il database di riferimento è diverso da quello aperto).
Cliccando su visualizzazione foglio dati si può testare l'anteprima del risultato e si possono inserire dei dati da accodare.

‹5› ATTENZIONE IL CONTATORE STA PER ESSERE MODIFICATO

Cliccando sul tasto che esegue la macro, ovvero la query di accodamento, indicato con un punto esclamativo di colore rosso, si ottengono i risultati.
Access avverte (con due differenti finestre di dialogo) che si tratta di una operazione irreversibile e che quindi non sarà possibile ripristinare la situazione precedente a quella generata dall'esecuzione della query di aggiornamento.

‹6› IL PROCESSO È COMPLETO. IL CONTATORE È AZZERATO

Dopo aver inserito un nuovo record questo verrà accodato sulla tabella originale (in figura si vede la seconda esecuzione).
Per sicurezza dalla modalità struttura della tabella Libro si può eliminare e subito dopo ricreare il campo IDL. Nell'inserire nuovi dati il contatore riparte dal 1, esattamente come ci aspettavamo.
Al termine è necessario eliminare la tabella di supporto per poter ricominciare da una base di dati pulita da tabelle ridondanti.

Delphi 2005 rivoluziona lo sviluppo per Windows

Da grande, voglio fare l'Architect!

Le novità introdotte in Delphi 2005 sono veramente tante. Un'anteprima sulla distribuzione più ricca, l'Architect. Realizziamo con ECO II un'applicazione per gestire dati senza scrivere una riga di codice

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

MANUALI
A partire da fine gennaio 2005 sarà possibile prenotare presso i rivenditori le copie dei manuali di Delphi 2005 in italiano!

Non è facile parlare di Delphi 2005 senza esaltarsi, ma non può essere altrimenti perché ciò che ora si chiama Delphi 2005, non è altro che l'ultima evoluzione del noto, speriamo non solo ai più vecchi, Turbo Pascal.
Quando fu concepito Turbo Pascal era avveniristico in confronto ai prodotti concorrenti e ora, alla fine del 2004, Borland continua su quella strada presentando l'innovativo Delphi 2005.
La ricetta utilizzata da Borland all'apparenza è semplice:

- mettere insieme Delphi 7, Delphi 8 e C#Builder;
- puntare con decisione su Microsoft .NET, senza dimenticare Win32;
- migliorare ed omogeneizzare il tutto con cura e senso pratico;
- servire su un'interfaccia innovativa e semplice da usare.

Il risultato è uno strumento che permette di realizzare applicazioni native per Win32 e Microsoft .NET utilizzando i linguaggi Delphi e C#.
Ciò è insufficiente per comprendere a fondo questa nuova versione; l'unico modo per farlo consiste nel capire Borland stessa. Due anni fa Borland decise di cambiare il proprio approccio allo sviluppo del software spostando il baricentro della propria offerta dagli strumenti di sviluppo alla gestione del ciclo di vita delle applicazioni, in inglese Application Lifecycle Management (ALM). Per farlo acquisì BoldSoft, Starbase e TogetherSoft, perché avevano sviluppato strumenti per la progettazione, per la gestione dei requisiti e dei gruppi di lavoro. In questo modo Borland si preparava ad offrire agli sviluppatori tutti gli strumenti necessari ad automatizzare le fasi dello sviluppo di un software: raccolta dei requisiti, analisi, progettazione, sviluppo, test e distribuzione.
Delphi 2005 incarna alla perfezione questo nuovo corso, facendoci intuire anche i possibili sviluppi futuri.

LE NOVITÀ

Le novità introdotte in Delphi 2005 sono molte e alcune di esse richiederebbero un articolo dedicato.
L'interfaccia è quella già vista in Delphi 8 e C#Builder, frutto del progetto Galileo; alcune funzioni sono nuove, altre sono state migliorate e in alcuni casi fuse assieme. Da evidenziare le *personalità multiple dell'IDE* che si adegua al tipo di progetto attivo. Per esempio se è attivo un progetto Win32, vengono nascosti i componenti per .NET. Analogamente il debugger attivo corrisponde alla piattaforma del progetto su cui si sta lavorando.
Un esempio di funzioni vecchie fuse assieme ed unite ad altre nuove è il *Structure Pane*. Esso presenta le caratteristiche dei precedenti Code Explorer e Object Tree View, unite a quelle della nuova funzione Error Insight che evidenzia in tempo reale gli errori di scrittura del codice senza dover ricorrere alla compilazione.
La *Tool Palette* dispone ora di una più potente funzione di ricerca; quando è attivata, sono

visualizzati i gruppi ed i componenti che iniziano per la chiave digitata. Finora il posizionamento dei componenti sulle form si eseguiva con una operazione di clic&clic; nell'attuale versione della Tool Palette è stato aggiunta anche la possibilità di effettuare un vero *drag&drop*.
Un altro esempio di potenziamento degli elementi dell'interfaccia si può osservare nell'*Object Inspector*, che oltre a permettere la modifica degli attributi di un componente e l'assegnazione del codice da eseguire al verificarsi di un specifico evento, permette anche l'ispezione delle proprietà dei file facenti parte del progetto e visualizzati nel Project Manager.
Miglioramenti sono presenti anche nella *funzione di ricerca* all'interno dei file con la possibilità di visualizzare il risultato raggruppato per file.
Un aggiornamento importante è il supporto da parte di ogni elemento dell'IDE del formato UTF-8.
L'*IDE Error Reporting* è il sistema di cui è stato dotato Delphi 2005 per inviare direttamente a Borland, via Internet e previa conferma da parte dell'utilizzatore, le informazioni necessarie per determinare le cause dell'errore verificatosi e gli interventi da eseguire per eliminarle.
Utile la funzione per importare o *esportare* da e verso Visual Studio i progetti C#.

LE NOVITÀ DELL'EDITOR

Un discorso a parte deve essere fatto per l'editor che presenta tante novità e migliorie da non avere più molto in comune con i predecessori.
Le novità riguardano la presenza della *funzione di refactoring*, della SynEdit, del già citato Error Insight, dell'Help Insight e dell'History Manager, solo per citare le più importanti.
Il refactoring permette di rinominare i simboli identificatori delle variabili e dei metodi all'interno dell'intero progetto attivo. Permette anche, solo con il linguaggio Delphi, di convertire le stringhe presenti nel codice in resourcestring. Molto comoda è la possibilità di estrarre una porzione di codice trasformandola in modo furbo in un metodo da richiamare in casi simili. La *funzione SyncEdit*, similarmente a quella di refactoring, permette di rinominare i simboli, ma solo all'interno del blocco di codice selezionato. La funzione *Error Insight* segnala gli errori presenti nel codice in due modi: sottolineando con una linea rossa ondulata le parti di codice non corrette, visualizzando sotto un apposito nodo del Structure Pane gli errori riscontrati. Entrambe le segnalazioni spariscono alla risoluzione del problema. Posizionando il puntatore del mouse sopra un simbolo, la funzione di Help Insight visualizza una finestrella contente le informazioni sull'oggetto e la possibilità di visualizzare direttamente l'aiuto della guida in linea tramite collegamenti ipertestuali. Completamente nuovo l'History Manager. Senza dover ricorrere a strumenti esterni, conserva traccia di tutte le modifiche apportate al codice, permettendo il confronto fa due versioni e il ripristino di quelle precedenti. Se è usato insieme a StarTeam vengono esaltate le funzioni già citate con in più la possibilità dell'uso concorrente e di una più sofisticata gestione delle versioni. Anche il linguaggio Delphi è stato arricchito con alcune nuove istruzioni: il ciclo for..in, i tipi annidati e le funzioni inline (solo Win32). Ciò si è reso necessario per favorire meglio il rapporto tra progetti Win32 e .NET (migrazione e coesistenza), ma anche le prestazioni (funzioni inline).

INTERVENTI IMPORTANTI SUL FRONTE DELLA GESTIONE DEI DATI

Attraverso i BDP (Borland Data Provider) lo sviluppatore .NET può avere i dati "vivi" all'interno delle form ma anche l'accesso a dati distribuiti, oltre ad un'omogeneizzazione di interfaccia verso i database. dbGo ora *supporta sia ADO sia ADO.NET*. Il BDE esiste ancora, ma solo per tabelle in formato Paradox e dBase. Il Data Explorer è lo strumento per interagire con i diversi database e le basi di dati presenti; esso può essere usato per creare, modificare ed eliminare tabelle senza lasciare l'IDE di Delphi 2005. È stata completamente riscritta la funzione di migrazione dei dati che ora può essere eseguita anche su una singola tabella. I componenti che eseguono fisicamente la migrazione sono disponibili nella Tool Palette per essere liberamente utilizzati all'interno delle proprie applicazioni.
Rilevante la possibilità tramite il Data Explorer di verificare il corretto funzionamento delle stored procedure.
Due strumenti di analisi sono presenti in Delphi 2005: Cristal Reports for Borland Delphi da Business Objects per le applicazioni

GLI INTERVENTI APPORTATI A DELPHI 2005 HANNO INTERESSATO TUTTE LE AREE E...

Ultima Ora! Borland ha ufficialmente comunicato che C++Builder continuerà ad esistere come parte di Delphi. Non è ancora stata deciso quando; potrebbe essere rilasciato come aggiornamento dell'attuale Delphi 2005 oppure fare parte della prossima versione ufficiale.

NOTA

NUOVO EDITOR

Delphi 2005 possiede un nuovo editor più potente e allo stesso tempo più semplice. Refactoring, SyncEdit e Error Insight sono alcune delle nuove funzioni messe a disposizione degli sviluppatori per semplificare la scrittura del codice.

NUOVI STRUMENTI DI REPORTING

Cristal Reports for Borland Delphi da Business Objects e Rave Reports Borland Edition da Nevrona Designs sostituiscono definitivamente Quick Report. Il primo dedicato alle applicazioni .NET, il secondo a quelle VCL sia Win32 sia .NET.

.NET, Rave Reports Borland Edition da Nevrona Designs per le applicazioni VCL Win32 e VCL .NET.

ECCO ECO

Finora sono state descritte tutte le novità tipiche di uno strumento di sviluppo evoluto, ma non abbiamo ancora parlato di ciò che rende migliore e diverso Delphi 2005 rispetto alle versioni precedenti e alla concorrenza.
In pieno rispetto della nuova filosofia ALM di Borland, Delphi 2005 dispone di una nuova versione di ECO. ECO (Enterprise Core Objects) giunge alla seconda versione presentandosi più potente e versatile del suo predecessore. Oltre alla possibilità di realizzare applicazioni client/server, è possibile realizzare anche quelle multi-tier; supporta ASP.NET e permette la mappatura di basi di dati esistenti generando lo schema UML oppure le classi per gestire le singole tabelle. ECO II supporta ora ECO Space multipli; un ECO Space è lo spazio virtuale in cui gli oggetti appartenenti al modello vengono creati a runtime per poter essere manipolati. ECO II può essere utilizzato solo per sviluppare applicazioni web o desktop per la piattaforma .NET. Per meglio comprendere la potenza di ECO II vedremo con un esempio come sia possibile in pochi minuti realizzare un'applicazione che accede ad una base di dati, senza scrivere una sola riga di codice.

UN'ESEMPIO D'ECO

Iniziamo creando un nuovo progetto in C# basato sul template ECO WinForms Application e gli assegniamo un nome a piacere. Visualizziamo l'Eco Space del progetto ed aggiungiamo un oggetto di tipo PersistenceMapperBdp; la proprietà SqlDatabaseConfig è impostata dal sistema a <<*EMPTY PERSISTENCE MAPPER CONFIG*>>. Questo oggetto crea un collegamento tra l'Eco Space e una connessione.
Passiamo al Data Explorer e attiviamo, se non lo è già, la connessione alla base di dati di esempio Northwind presente su SQL Server. Con un drag& drop dal Data Explorer all'Eco space, creo una connessione (bdpConnection1) alla base di dati North wind; automaticamente la proprietà Connection dell'oggetto persistenceMapperBdp1 viene impostata a bdpConnection1.
Se posizioniamo il cursore del mouse sopra l'oggetto persistenceMapperBdp1, Delphi 2005 visualizza un riquadro giallo contenente un promemoria sulle azioni svolte e quelle da svolgere.
Utilizzando il menu contestuale di persistenceMapperBdp1 scegliamo il comando SQL Server setup; in questo modo Delphi 2005 recupera le informazioni necessarie a connettersi a SQL Server (proprietà SqlDatabaseConfig).
Dal menu contestuale dell'Eco Space eseguiamo il comando Wrap Existing Database with ECO per generare il diagramma UML della base di dati. Se andiamo nel Model View e scegliamo il package NorthwindClassesPackage, con un doppio clic sul riferimento allo schema è possibile visualizzare il diagramma UML della base di dati. In questo modo si riesce ad osservare tutte le relazioni presenti tra le tabelle. Se il diagramma è affollato è possibile chiedere allo strumento, con il comando La-

UN PROGETTO SENZA CODICE

CREIAMO UN'APPLICAZIONE C#

1 **Con il comando File/New/Other generiamo una nuova applicazione C# basandola sul templete ECO WinForms Application.**

IMPOSTIAMO L'ECO SPACE

2 **Aggiungiamo all'Eco Space i componenti necessari a connettersi al database e li colleghiamo tra loro.**

GENERIAMO IL DIAGRAMMA UML

3 **L'Eco Space genera il diagramma UML della base di dati che abbiamo scelto, completo delle relazione tra tabelle.**

yout/Do Full Layout del menu contestuale, di disporre meglio gli oggetti del grafico.
Salviamo per non perdere il lavoro svolto finora. Sempre all'Eco Space aggiungiamo un oggetto di tipo FileMappingProvider; in questo modo diciamo all'Eco Space quale file di mapping utilizzare. Alla proprietà FileName di fileMappingProvider1 assegno il valore NorthwindMapping.xml completo di percorso. Assegno alla proprietà RunTimeMappingProvider di persistenceMapperBdp1 il valore fileMappingProvider1. Dal menu contestuale dell'Eco Space lanciamo il comando Select Packages; nella finestra di dialogo visualizzata sostituiamo il package di default con quello appena creato. Salviamo, compiliamo ed eseguiamo per prova; apparirà una finestra vuota. Una finestra vuota non serve a nulla, abbiamo bisogno quindi di un oggetto di tipo DataGrid ed uno di tipo ExpressionHandle. A quest'ultimo componente assegniamo un nome più significativo (ehProduct perché punterà alla tabella dei prodotti) e impostiamo la proprietà RootHandle a rhRoot. Ritorniamo sulla nostra form e impostiamo la proprietà Expression dell'oggetto ehProduct scegliendo la tabella e le colonne; nell'apposito riquadro della finestra di dialogo dell'editor di proprietà deve apparire la stringa "Product.allInstances".
Selezioniamo la griglia per impostare la proprietà DataSource affinché punti al corretto expression handle: ehProduct.
Impostiamo la proprietà EcoAutoForm della DataGrid a True, affinché con un doppio clic su una riga della tabella visualizzi una autoform Eco per modificare i dati e navigare tra le relazioni. A questo punto abbiamo finito e possiamo mandare in esecuzione e navigare sulla tabella dei prodotti e sulle tabelle ad essa collegate.

CONCLUSIONI

Un Delphi così con lo avevamo mai visto.
Forse la stessa sensazione l'abbiamo provata in occasione delle uscite di Delphi 1 e Delphi 2, perché rispettivamente rappresentavano la nascita ed il passaggio al mondo a 32bit.
È multi linguaggio, è multipiattaforma, abbraccia l'ALM, si integra con CaliberRM e StarTeam. In poche parole si candida a diventare lo Strumento di sviluppo per Windows.
In questa versione tutti linguaggi e le piattaforme supportati guadagnano qualcosa. Chi perde qualcosa per strada e la parte Win32, perché non troviamo più Bold for Delphi e ModelMaker (introdotti nella versione 7),e QuickReport. Bold for Delphi è stato geneticamente modificato per passare da Win32 a .NET e ora si chiama ECO. ModelMaker non viene più distribuito da Borland ma si può acquistare a parte direttamente dal produttore. QuickReport, qualcuno esclamerà "finalmente", è stato sostituito da Cristal Reports e da Rave Reports, che era già presente in Delphi 7 in previsione dell'eliminazione di QuickReport. Molti lo avevano già sostituito con prodotti di terze parti. Queste assenze non devono però essere viste come un difetto del prodotto, ma piuttosto come la prova della lungimiranza di Borland, perché Windows sta puntando dritto su un mondo .NET e Borland non solo non vuole rimanere indietro, ma vuole essere uno dei protagonisti nello scenario che Microsoft sta allestendo per Windows e le tecnologie collegate.

Andrea Böhm

SUL WEB

Esistono sul Web molte fonti di informazioni sui prodotti Borland dagli ufficiali, passando per quelli dei partner, per arrivare a quelli indipendenti. Non potendo citarli tutti di seguito una selezione:

Borland Italia:
http://www.borland.it
Borland Developer Network (articoli):
http://bdn.borland.com
Borland Code Centrale (esempi):
http://cc.borland.com/ccweb.exe
Borland Blogs:
http://blogs.borland.com
Torry's Delphi (componenti e altro):
http://www.torry.net

AGGIUNGIAMO UNA GRIGLIA

4 Per visualizzare i dati utilizzeremo una DataGrid, un componente standard di .NET e la colleghiamo alla tabella prescelta.

L'APPLICAZIONE FUNZIONANTE

5 Una volta lanciata, l'applicazione visualizza immediatamente i dati della tabella nella griglia.

LE TABELLE COLLEGATE

6 Un doppio clic su una riga della griglia e, sfruttando gli autoform di Eco, possiamo navigare tra le tabelle collegate a quella scelta.

XML, XSL, PDF, HTML, CSV, tutto 100% Java!

Creare report in Java con Jasper

Per venire incontro alle esigenze di reportistica delle vostre applicazioni, vi presentiamo Jasper. Un potente progetto open-source con cui aggiungere al software Java il potere della sintesi!

Non c'è applicazione che non abbia bisogno di salvare dati. È per questo motivo che le tecnologie per la gestione e l'utilizzo dei database sono così importanti nell'imparare a programmare. In maniera del tutto analoga, posso estendere l'affermazione al punto di dire che quando abbiamo dati in un'applicazione, avremo sicuramente anche bisogno di estrapolare statistiche, sintesi e report su quei dati, operazioni che possiamo catalogare complessivamente come reportistica. Uno dei linguaggi che va per la maggiore al giorno d'oggi è sicuramente Java: gratuito, universalmente disponibile, ed utilizzabile con una serie di ambienti integrati di sviluppo a loro volta gratuiti ed open-source (in primis, Eclipse e NetBeans). Vi presentiamo Jasper: si tratta di un progetto open-source per cui dobbiamo essere grati al mondo Java e la sua funzione è quella appunto di permettere di formattare svariate tipologie di contenuto su schermo, stampante o file di tipo PDF, HTML, XLS, CSV e XML. Il principio di questo prodotto è molto semplice: un file XML di input rappresenta un template di design del report che vogliamo generare, mentre con le classi dell'API di Jasper possiamo generare degli output creando dei documenti il cui layout e formato sono definiti nel template XML ed i dati sono estrapolati da fonti che tipicamente sono database con driver JDBC, ma che il livello di astrazione di Jasper rende infinite.

I TEMPLATE DI JASPER

Come si è già accennato, il template è un modello in base al quale i dati da visualizzare saranno formattati ed organizzati sulla pagina di presentazione (sia essa su video, stampante o file). Abbiamo già detto che si tratta di file XML, i quali seguono un DTD ben specifico e la cui intestazione ha questa forma:

DEFINIAMO UN TEMPLATE

1 **Menu *Datasource -> Connection/Datasources*, pulsante *New:* impostiamo i parametri di accesso ad una fonte dati JDBC e attiviamola tramite il menu *Build->Set active connection***

2 **Menu *File->Report Wizard*, imposto la stringa SQL per il recupero dei dati dalla fonte impostata e resa attiva al passo 1**

```
<?xml version="1.0"?>
<!DOCTYPE jasperReport PUBLIC
                "-//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport name="NomeDelReport" ... >
...
</jasperReport>
```

All'interno di questo file XML abbiamo poi a disposizione una vasta serie di tag che ci permettono di creare tutti quegli elementi necessari alla definizione di un report professionale. Innanzitutto il nostro documento consterà di una serie di sottoelementi per l'organizzazione visuale (pagine, intestazioni, piè di pagina, sfondo, titolo, colonne) e di aggregazione dati (gruppi, con relative intestazioni e piè di pagina, sommari). Il tag parameter permetterà di definire parametri con cui ottenere informazioni dall'ambiente esterno al momento della generazione del report (quando cioè i dati vengono elaborati, tramite il template, per produrre la presentazione di output). I campi rappresentano proprio i dati che il nostro report deve visualizzare e che vengono estrapolati da una fonte definita a run-time (a cui non si fa quindi riferimento nel file XML di template) che tipicamente è un database cui si accede via JDBC. Per rendere i report più flessibili e strutturalmente vari, Jasper offre anche la possibilità di creare delle variabili all'interno dei report il cui valore viene calcolato ad opera di espressioni che portano tutta la potenza del linguaggio Java all'interno del template. Tali variabili possono operare sui campi, sui parametri o su alcuni valori messi a disposizione dal prodotto a run-time (numero di pagine, nome della sezione, numero di colonne, etc). Infine, un potentissimo strumento di Jasper sono gli scriptlet, che possiamo descrivere come degli eventi lanciati prima e dopo determinate operazioni del motore di report. Per utilizzare gli scriptlet è necessario esplicitare l'attributo *scriptletClass* dell'elemento *root* del template di JasperReport. Il valore da dare all'attributo corrisponde al nome della classe Java che conterrà le operazioni da compiere. La classe può essere derivata da o *JRAbstractScriptlet* oppure da JRDefaultScriptlet e i metodi che esporrà sono: *beforReportInit*, *afterReport-*

IL SITO DI JASPER

JasperReports è presente su internet all'indirizzo *http://jasperreports.sourceforge.net*, dove trovate il link per scaricare il prodotto ed alcune informazioni relative al suo utilizzo, oltre a link esterni a tool GUI che sfruttano la libreria e la rendono di più facile utilizzo.

PER JASPER REPORT CON IREPORT

3 Seleziono i campi che devono essere inclusi automaticamente nel report ed associati a campi (field) del template

4 Creo delle eventuali aggregazioni selezionando i campi della tabella in base ai quali desidero raggruppare i dati

5 Seleziono uno dei layout predefiniti per il mio report. Il template generato potrà ovviamente essere personalizzato in seguito. Dopo una finestra di riepilogo, clicco sul pulsante Finish e viene generato il mio template

6 Risultato di esecuzione (seconda pagina) di un template generato con il report wizard

DTD

Il DTD (Document Type Definition) è un file con cui si dà una definizione di nuovi tipi di documento XML, dichiarando quali tag sono ammessi, in che ordine, con quale nidificazione e con quali attributi.

Init, *beforePageInit*, *afterPageInit*, *ecc*. Gli script non saranno altro che il contenuto dei metodi, mentre appare evidente che i nomi dei metodi fanno riferimento al momento in cui saranno eseguiti.

DISEGNARE UN REPORT

Siccome risulta molto difficile dare un layout in termini di tag XML ad un design visuale, di solito per la creazione dei template si fa ricorso a tool visuali che avvalendosi delle tecnologie di drag-and-drop e di svariate finestre di proprietà ci nascondono parte dei dettagli di implementazione di Jasper e della struttura DTD sottostante. Il sito ufficiale di Jasper Report offre una lista di tali strumenti, ma molti di essi sono commerciali e vanno contro il nostro principio di parlarvi per quanto possibile di strumenti accessibili a tutti (soprattutto per tool che sfruttano tecnologie a loro volta open-source e gratuite): ne ho così selezionato uno solo tra i vari proposti che, pur essendo completo e semplice, si propone al pubblico a costo zero. Il nome del progetto è *iReport* ed il suo sito ufficiale è *http://ireport.sourceforge.net* (ne trovate comunque una copia della versione 0.4.0 sul CD allegato, visto che si tratta di ben 11Mb). Creare dei template con *iReport* è veramente molto semplice. Potete partire da un documento vuoto ed aggiungere graficamente i vari elementi associandoli alla definizione dei campi della tabella che volete visualizzare, od utilizzare il Report Wizard per cominciare con un report già pronto da modificare.
Per utilizzare il wizard dovete innanzitutto creare una fonte dati e renderla attiva (cioè utilizzata per default dal prodotto).
Le **Digure** da **1** a **6** vi mostrano i passi da seguire per avere un report pronto e funzionante.

COME INSTALLARE JASPER E IREPORT

Per poter utilizzare JasperReports nelle vostre applicazioni è sufficiente scaricarne il JAR dal sito del prodotto (l'ultima versione è la 0.6.2) ed aggiungerlo al classpath della macchina virtuale dentro cui le applicazioni girano. Per utilizzare iReport, invece, richiamate semplicemente il comando .bat della distribuzione – iReport infatti già include la versione 0.6.1 di JasperReports.

COME SI USA UN TEMPLATE

Anche se iReport si occupa, come abbiamo detto, del ciclo completo dell'utilizzo di Jasper, dall'ideazione grafica sino alla compilazione del template e alla generazione dei report stessi, prima di lasciarvi alle vostre sperimentazioni mi premeva darvi alcune informazioni sulle procedure di basso livello per l'utilizzo di Jasper nel codice Java, per poter così dotare le applicazioni che scrivete di un sistema di sintesi e statistica delle informazioni indipendente da tool esterni quali iReport. Per risparmiare sullo spazio, avviso in anticipo che tutte le classi che citerò fanno parte del package *net.sf.jasperreports.engine* della distribuzione di Jasper. Una volta creato il template XML, la classe *JasperCompileManager* è in grado di "compilare" tale file verificandone la correttezza formale e semantica e trasformandolo in un'istanza di *JasperReport*. I metodi per operare tale trasformazione sono diversi, per permetterci di usare file, stream o instanze serializzate in input e produrre file, stream o istanze di oggetti in output: quello che userete probabilmente di più è comunque *compileReport* che, dato il nome di un file XML di template, vi crea un *JasperReport* da usare nel vostro codice. Il secondo step è passare dei dati al report per poter generare una visualizzazione basata sul template appena compilato. A questo scopo entra in gioco la classe *JasperFillManager,* il cui compito è quello di prendere un *JasperReport* ed una connessione ad una fonte dati, e generarvi un'istanza di *JasperPrint*. Anche qui, il metodo più popolare sarà con certezza *fillReport,* che in sequenza vuole:

- il *JasperReport* ottenuto da *compileReport*
- una Map con i valori dei parametri dichiarati nel template
- un riferimento ad una fonte dati, che può essere una connessione JDBC oppure una generica fonte *JRDataSource,* sicuramente più flessibile ma meno immediata.

INFINE L'OUTPUT

Come passo conclusivo, non vi resta che scegliere cosa fare della presentazione generata dall'unione del template con i suoi dati: potete visualizzarlo su schermo invocando il metodo *viewReport* della classe *JasperViewer* (package *net .sf.jasperreports.view*) cui passate un *JasperPrint,* oppure utilizzare la classe *JasperExportManager* ed i suoi metodi *exportReportToHtmlFile, exportReportToPdfFile, exportReportToXmlFile,* che vogliono un *JasperPrint* ed il nome del file su cui scrivere (per i file XML dovete anche indicare se volete includere i byte di eventuali immagini nel file stesso o referenziarli esternamente). Se infine volete ottenere un report in XSL o CSV, potete utilizzare le classi *JRXlsExporter* e *JRCsvExporter:* esse non sono gestite ed utilizzate direttamente da *JasperExportManager,* ma fanno parte della stessa tipologia funzionale. A questo punto, direi che avete a disposizione tutto il necessario per creare report in Java ed ora, finalmente, tocca a voi prendere il controllo di *JasperReports!*

Federico Mestrone

SOFTWARE SUL CD

INTERNET

Apache 1.3.33/2.0.52

Uno dei server Web più usati al mondo

Ormai indispensabile, ioProgrammo lo ripropone spessissimo fornendovi sempre le versioni più aggiornate. In questo numero Apache sarà usato per gestire più di un progetto. Si va da EZPublish a Tomcat, e chiaramente all'integrazione con PHP. Se avete bisogno di un server Web per provare le vostre applicazioni Web, oppure da usare in sistemi di produzione, Apache è quello che fa per voi.

Directory /Apache

EZPublish 3.1.4

Un sistema di sviluppo per CMS

EZPublish non è il solito Content Management System. Non vi mette a disposizione un tool predefinito con il quale creare i vostri contenuti OnLine. EZPublish è un Framework per la generazione di CMS. Con EZPublish potete creare i vostri oggetti, specificandone attributi e caratteristiche, per poi gestirne il layout tramite semplici template HTML. In questo numero vi offriamo la versione con Installer che, tramite una comoda procedura guidata, installerà EZPublish sul vostro computer senza troppi problemi.

Diretory /ezpublish

PHP ADS New 2.0

Gestisce in modo profesionale le campagne banner su Internet

Un grande strumento che consente di vendere ai vostri clienti campagne banner monitorate. Conteggia le esposizioni e i click, consente di gestire la priorità dei banner, permette di gestire le "Zone", ovvero consente di targettizzare al massimo l'esposizione dei banner. E ancora molto altro, come ad esempio la possibilità di stabilire KeyWords o priorità di esposizione, e naturalmente gestisce sia campagne a tempo che per click. Davvero professionale!

Directory /PHPAdsNew

NucleusCMS

Probabilmente il primo CMS orientato al blogging

Nucleus ha forse aperto la strada al Blog. Si tratta di una web application in PHP per l'immissione rapida di contenuti sul Web. Come piattaforma è molto simile a quella del blogging, data la sua anzianità è dotato di una serie straordinaria di Plugin, è inoltre molto flessibile, semplice da usare e configurare

Directory /Nucleus

PHPWiki

Ottimo per la creazione di documentazione scritta in modo collaborativo

Si tratta di uno strumento aperto per la creazione di documentazione. In sostanza, chiunque può modificare o aggiungere un contenuto. Viene mantenuta una trac-

PHPMyAdmin 2.6.0

Molto probabilmente il Frontend più usato al mondo per MySQL

Ci possono essere un milione di motivi per cui usare un'applicazione Web come frontend verso MySQL, ad esempio quando il vostro provider supporta solo la connessione a localhost è assolutamente indispensabile. PHPMyAdmin tuttavia non è un ripiego a un'interfaccia grafica evoluta, anzi la complessità delle funzioni che esporta lo rende molto spesso preferibile a un'interfaccia standalone.

Directory /PHPMyAdmin

1 LANCIAMO PHPMYADMIN - Dopo averlo installato, è possibile accedere a phpmyadmin da http://localhost/phpmyadmin. Se avete installato PHPMyAdmin su un server remoto, sarà necessario sostituire a localhost il nome dell'host che ospita il server e a phpmyadmin il percorso corretto.

2 CREIAMO UN DATABASE - È sufficiente indicare il nome del database nell'apposita textbox e poi cliccare su 'crea'. PHPMyadmin creerà per voi la struttura di un database vuoto. La pagina successiva vi inviterà a popolarlo con tabelle e con dati.

3 POPOLIAMO IL DATABASE - Nel nostro caso ci limiteremo a importare la struttura e i dati da un file sql preesistente. Per farlo, selezionate la 'tabsheet SQL'. Nella pagina che segue utilizzate il tasto 'Browse' per selezionare il db da importare, e infine cliccate su 'Esegui'

SwishLite

Questo mese vi presentiamo in versione completa un tool che consente di creare piccole applicazioni e filmati Flash dotati di effetti straordinari con un impegno minimo

SwhishLite è un grande prodotto. Dedicato ai programmatori e ai grafici, specializzati in campi diversi da quelli dell'animazione vettoriale, non vogliono comunque perdere la possibilità di rendere esteticamente accattivanti le proprie applicazioni. ShishLite consente di realizzare effetti sorprendenti, offre un tool minimo ma molto bilanciato di funzionalità che consentono di interagire e personalizzare i filmati, e infine consente di esportare in formato SWF. Esiste, naturalmente anche la possibilità di importare filmati realizzati con Flash per una successiva elaborazione. Insomma si tratta di un prodotto completo che può aiutarvi in tutte quelle situazioni dove la vostra vena artistica non è sufficiente a supportare le vostre capacità di programmatore. Dal punto di vista della programmazione intesa in senso stretto supporta tutti gli eventi di gestione del mouse, e fornisce una serie di azioni possibili predefinite che rappresentano un subset delle fondamentali di Macromedia Flash. È molto completo dal punto di vista della realizzazione rapida di effetti, e ci sono centinaia di combinazioni possibili per dare sostanza alla vostra fantasia. Si va dalla gestione dell'alpha, al movimento, allo scaling, a tutte quelle accortezze grafiche che consentono di creare effetti sorprendenti in tempo brevissimo.

UN FILMATO IN QUATTRO TEMPI

1 All'avvio del programma, l'interfaccia si presenta come in figura. Progettiamo un filmato di dimensioni 480x270 pixel. Inoltre impostiamo uno sfondo nero.

2 Settiamo il numero di fotogrammi del nostro filmato a 100. Inoltre cliccando sull'icona posta sulla barra degli strumenti, aggiungiamo un testo.

4 Definiamo il contenuto che deve essere proiettato nel filmato, il suo allineamento, i colori e quant'altro riguarda il posizionamento del testo nel filmato.

5 Cliccando con il tasto destro sull'etichetta del testo scegliamo uno dei tanti effetti disponibili. Per visualizzare il risultato clicchiamo sul tasto.

INSTALLAZIONE

Nel nostro CD nella directory SwishLite trovate l'eseguibile "SwishLite.exe". Per ottenere il codice di sblocco che vi abiliterà alla versione completa si SwishLite dovrete registrarvi come clienti all'indirizzo www.swishzone.it/register, nel campo codice prodotto dovete inserire il codice 6874267196. Dopo qualche giorno vi arriverà via email il codice di sblocco che vi consente di abilitare definitivamente SwishLite.

SWHISMAX

Allegato a questo numero di ioProgrammo trovate SwishMax, una versione molto potenziata rispetto a SwishLite.
Quasi un vero ambiente di programmazione, ma ancora una volta contraddistinto dalla facilità di utilizzo. In SwishMax alcune delle differenze sono:

- **Oltre 330 effetti interni;**
- **Possibilità di poter distribuire gli effetti creati;**
- **Supporto SWiSHScript con script editor e debugger interni;**
- **Supporto per inserimento di campi per la creazione di form;**
- **Opzioni avanzate di importazione ed esportazione;**
- **Opzioni avanzate per la creazione di shape tra cui: Linea, Matita, Bezier, Testo, Ellisse/Cerchio, Rettangolo/Quadrato e AutoShape per oggetti 3D;**

Il costo di SwishMax è irrisorio se paragonato alle potenzialità. Si parte da 119.95 € e ci sono offerte vantaggiose anche per chi aggiorna da SwishLite.
Decisamente da provare.

cia delle modifiche e quelle autorizzate contribuiscono alla creazione delle documentazione.
Directory /PHPWiki

STRUMENTI

Jcreator 3.1.0

Un grande editor per Java

Si tratta di un IDE strepitoso ma molto leggero. Dotato di tutte le caratteristiche di un editor di alto livello, dall'highLighting sintattico, alla vista ad albero della gerarchia del progetto, mantiene in ogni caso la sua leggendaria leggerezza. In un paio di mega vengono racchiuse comunque tutte le funzionalità tipiche di ambienti molto più pesanti
Directory /Jcreator

PHPEclipse

Un plugin per sviluppare applicazioni PHP con Eclipse

Eclipse è un progetto che sta riscuotendo un enorme successo, soprattutto in ambito Java. Si tratta, come molti sapranno, di un completo ambiente di programmazione per Java. Tuttavia, con questo Plugin, Eclipse diventa un editor molto evoluto per PHP, dotato di complection del codice, sintax highlighting, integrazione con Apache e Mysql. Un vero e proprio ambiente evoluto dal costo praticamente nullo. L'installazione è semplicissima. È sufficiente decompattare i file di PHPEclipse nella directory dei plugin di eclipse ed avviare l'ambiente. Cliccando su "File/New Project" vi ritroverete un ramo "PHP" che contiene "PHP Project", questo ci consente di avviare un nuovo progetto PHP.
Directory /PHPEclipse

MySQL Connector .NET

Il connector per usare MySQL direttamente da .NET

A questo strumento abbiamo dedicato un intero articolo in ioProgrammo 87. È un connettore che consente di accedere a DB MySQL direttamente da .NET utilizzando metodi nativi e senza passare da ODBC. Ovviamente questo aumenta moltissimo le prestazioni, inoltre il basso costo di MySQL e le sue performance degne di SQL Server e talvolta superiori lo rendono un ottima alternativa per lo sviluppo di qualunque progetto che faccia uso di Database.
/Mysql/mysql-connector-net-1.0.2-gamma.zip

EgroupWare V1.0

Incredibilmente potente per la gestione di gruppi di lavoro

L'installazione è molto semplice. Ovviamente dovete avere installato il trio Apache+MySQL+PHP. Grazie al nuovo tool sotto Windows, egroupware verrà installato in una qualunque cartella, di solito la *:/Programmi/egroupware-1.0.00.006/egroupware.* A questo punto è necessario aggiungere qualche riga nel *httpd.conf* di apache, in particolare:

```
Alias /egroupware "C:/Programmi/egroupware-1.0.00.006/egroupware"
<Directory "C:\Programmi\egroupware-1.0.00.006/egroupware">
   Options +ExecCGI +FollowSymLinks
 AllowOverride Limit
 Options Indexes
 DirectoryIndex index.php
</Directory>
```

Infine puntate il vostro browser verso http://localhost/egroupware. Se tutto è andato a buon fine partirà la procedura di installazione di Egroupware
Al termine avrete a disposizione un completo sistema di gestione dei gruppi di lavoro, con tanto di agenda condivisa, gestione dei progetti, file manager e tantissime altri strumenti.

Directory /egroupware

MySQL Control Center

L'interfaccia grafica per gestire DB Mysql

E' sicuramente il FrontEnd più utilizzato con database MySQL. Offre tutte le funzionalità per gestire un'intero Server DB. Dalla creazione di database, alla gestione degli utenti, al browsing delle tabelle, alla costruzione di Schema, all'immissione dirette di comandi SQL. Sicuramente un must per chi vuole amministrare un DB MySQL senza problemi.
Directory /Mysql/mysqlcc-0.9.4-win32.zip

LINGUAGGI

PHP 4.3.10/5.0.3

Il linguaggio di scripting per il Web

Ormai PHP lo conoscete tutti e, se non lo avete mai usato, sicuramente vi sarà capitato di accedere a qualche sito Web sviluppato con PHP. Saprete dunque che è un linguaggio di scripting particolarmente utilizzato per sviluppare Web Application. Incredibilmente potente, fa della completezza del linguaggio, della facilità di apprendimento, della capacità di integrarsi con applicazioni di Dababase i suoi punti di forza.
Directory /PHP

LIBRERIE

ASP

AZTEC codici a barra per ASP.NET

Tutto quanto necessita per un controllo completo nella produzione originale di codici a barre.

[MW6AztecASPNet.ZIP]

PDF417 BarCode ASP.NET

Qualità professionale nel generare codici a barre in due dimensioni.

[MW6PDF417ASPNet.ZIP]

Dall'IP al paese d'origine

Un modo semplice e veloce per avere informazioni su chi visita il vostro sito.

[scandportal_iptocountry.zip]

Compressione ZIP con ASP.NET

Per comprimere, decomprimere e criptare i dati in modo veloce e flessibile.

[ZipDotnet.msi]

DELPHI 2005

Ne parliamo diffusamente in questo stesso numero di ioProgrammo. Si tratta della nuova piattaforma di programmazione di Borland che, fra le novità più interessanti, aggiunge il supporto a .NET.

In parole povere è possibile sviluppare con Delphi, ovvero il primo ambiente RAD ad avere conquistato il mercato, utilizzando Object Pascal per la piattaforma .NET.
Chiaramente non si tratta dell'unica novità di questo ambiente. Ma creare la mia prima applicazione ASP.NET in Object Pascal con Delphi è stato emozionante come fu circa 10 anni fa creare la mia prima applicazione con un ambiente RAD quale era allora Delphi 1.0.

Directory /Delphi2005

Installazione

I requisiti per l'installazione di Delphi 5 sono principalmente avere installato una versione del .NET framework e l'MSXML Core Service SP2, ambedue scaricabili da *http://www.microsoft.com/download*.
Al solito, per problemi relativi alle licenze non siamo in grado di offrirvi questi strumenti sul CD allegato alla rivista, ma stiamo cercando di aggirare anche questo problema. È necessario anche avere una licence key del prodotto che consentirà di usarlo per trenta giorni.
Per ottenerla è necessario registrarsi presso *http://www.borland.com/products/downloads/download_delphi.html*, cliccando su "Delphi 2005 Architect Trial". Vi verrà spedita una licenza via email.
Dovete salvare il file che contiene la licenza sull'Hard Disk e importarlo poi con il licence manager che viene installato da Delphi 2005.
D'accordo tutto questo può sembrare una fatica immensa, ma in realtà stiamo parlando di pochi minuti.
Quello che poi potete fare con Delphi 2005 vi ricompenserà ampiamente.

Una prima WebApplication.NET

1 INIZIALIZZARE IL PROGETTO - Avviamo il wizard scegliendo *"NEW"* nella pagina che ci viene mostrata avviando Delphi5. Subito dopo scegliere *"Asp.NET Web Application"* da *"Delphi for .NET projects"*

2 IL NOME DELL'APPLICAZIONE - Inserite un nome significativo per l'applicazione e la sua posizione sull'HD. Se avete installato un Web server locale salvate i file in una directory gestita dal Web Server e indicatene il tipo

3 DISEGNIAMO L'APPLICAZIONE - Nella form che compare trasciniamo una *Label* e un *Bottone*, tutti e due dalla categoria *Web Controls*. Settiamo le proprietà della label determinando la dimensione del testo dall'object inspector.

```
procedure TWebForm1.Button1_Click(
        sender: System.Object; e:
                System.EventArgs);
begin
  label1.text:='hello world';
end;
end.
```

4 CODICE PASCAL - Cliccando due volte sul bottone che abbiamo messo sulla form si può programmare l'evento *OnClick* del bottone. Nel nostro caso molto semplicemente cambieremo l'etichetta della label in *Hello World*

Deployment dell'applicazione

L'applicazioncina di test che abbiamo creato è pronta per essere installata su un server Web di produzione che supporti ASP.NET. Possiamo farlo o cliccando su "file/new/other/Deployment e poi su ASP.NET Deployment", oppure direttamente via FTP. Dovete tenere conto pero' che la directory che ospiterà l'applicazione deve essere una Virtual Directory con i permessi d'esecuzione.

Python 2.3.4

Un linguaggio orientato agli oggetti con tanto di supporto a classi ed ereditarietà

Un linguaggio usato in tutto il mondo in una varietà di applicazioni, pare addirittura che venga utilizzato dai team di Google per lo sviluppo di alcune funzionalità lato server del miglior motore di ricerca esistente. Si caratterizza per la gestione dinamica della memoria, per l'elevata portabilità, per la curva di apprendimento relativamente breve. Un linguaggio di programmazione di cui sentiremo parlare a lungo.

Directory/Python

MySQL 4.1.7

Il server di database OpenSource più diffuso al mondo

MySQL è un'indispensabile. ioProgrammo ogni mese vi offre la versione aggiornata. Si tratta del server di database fondamentale

per la maggior parte delle applicazioni Internet scritte in PHP. Ma è difussissimo anche per le applicazioni Standalone e grazie ai nuovi connector comincia a essere usato anche dagli sviluppatori .NET
Directory /Mysql

TOOL DI SVILUPPO

Stylus Studio 6

Lo strumento più completo per la gestione di documenti XML

Un editor XML tra i più potenti e completi che possiate trovare. L'interfaccia è ricchissima di pulsanti e opzioni ma, grazie ad una organizzazione esemplare, riesce a non disorientare già dal primo utilizzo. La finestra di editing è organizzata a tab e consente dunque di passare rapidamente da un file all'altro; inoltre, per ogni file XML, è possibile scegliere fra quattro diverse visualizzazioni: *Text*, *Tree*, *Grid* e *Schema*.
Ancora più interessanti le opzioni di visualizzazione disponibili per trasformazioni XSLT. Anche in questo caso possiamo scegliere fra quattro possibilità di visualizzazione: XSLT Source, che mostra il sorgente; Mapper, che fornisce una efficacissima rappresentazione grafica delle relazioni fra documento di partenza e documento trasformato; Params, che offre una rappresentazione personalizzabile dall'utente; WYSIWYG, come intuibile dà una rappresentazione del documento come si presenterà nel browser dopo la trasformazione.
Anche verso i database Stylus Studio dispone di utili funzioni sia per l'import sia per l'export.
Al primo vi verranno chiesti un indirizzo mail e pochi dettagli anagrafici. In pochi istanti vi sarà inviata un chiave di attivazione valida trenta giorni.

studio.exe

Poseidon for UML Community Edition 3.0

Per creare diagrammi UML e fare il reverse engineering del codice Java

Poseidon è un software per la progettazione UML di livello professionale. La versione Community Edition che qui presentiamo è gratuita e consente sia la generazione di diagrammi UML sia il reverse engineering a partire da codice sorgente Java. Implementato completamente in Java, può girare su qualsiasi piattaforma. I diagrammi sviluppati con Poseidon possono essere esportati in svariati formati (gif, ps, eps e svg), pieno supporto per il drag & drop, interessanti funzionalità per il reverse engineering di sorgenti Java, generazione automatca di codice Java. Compatibile con lo standard UML 1.3, Poseidon supporta tutti i diagrammi UML.

LIBRERIE ASP

TEXTBOX e ASP.NET

HTML Box Control customizzabili in ogni aspetto e larga compatibilità.

[TextBoxDotnet.msi]

LIBRERIE C#

ChilKat E-Mail Component

Invio e ricezione di e-mail completo da provare per trenta giorni.

[ChilkatDotNet.msi]

WebCabStatistics

Versione 3.2 di un componente apprezzato per la varietà delle funzioni che rende disponibili.

[WebCabStatisticsDemoNETService.Msi]

Telnet Tool

Tutto quanto necessita per la gestione del protocollo telnet.

[TelnetTool.exe]

FTP DOTNET

Potente FTP Client per sopperire alla mancanza in questo ambito di .NET

[FtpDotNET.msi]

Trasmissioni sicure

Autenticazione e criptazione per una trasmissione sicura dei dati.

[SecureTool.exe]

TESTIAMO UNA QUERY XPATH CON STYLUS STUDIO 6

1 **Dal menu *File* selezioniamo la voce *Open* e indichiamo il percorso del file XML su cui voglia effettuare la query XPath.**

2 **Al di sopra della finestra di editing, selezioniamo la casella di testo in cui è presente la dicitura <*Type a new query*> e scriviamo la query XPath.**

3 **Premendo *Invio* otterremo, nella finestra collocata sulla sinistra, la lista dei nodi che soddisfano i criteri indicati nella query.**

SCOPRIAMO LA STRUTTURA DELLE CLASSI CON POSEIDON

1 Dal menu *File*, selezioniamo la voce *Import files*. Indichiamo il percorso del file .Java di cui vogliamo effettuare il reverse engineering.

2 Ci verrà presentato un diagramma ridotto del package cui appartiene il file che abbiamo indicato. Con un segno di spunta, possiamo indicare gli elementi da analizzare.

3 Possiamo esplorare una qualsiasi classe del nostro progetto. In basso a destra accediamo a tutte le proprietà ed al codice sorgente, sulla sinistra resta in evidenza la struttura del package.

LIBRERIE

LIBRERIE C++

dtSearch Text Retrieval Engine

Un "motore di ricerca" per rendere ancora più completa la tua applicazione.

[00023496.EXE]

PrintPRO

La potenza di un componente per la stampa avanzata (e distribuita).

[00020325.exe]

TwainPRO

Versione 4.0 di un componente specifico per l'acquisizione da scanner.

[00020459.exe]

BCGControlBar Library

Per creare applicazioni con una potente interfaccia in stile Microsoft Office.

[000222282.exe]

III Computing C++ Class Library

Una gran classe per C++ che aggiungerà molte funzionalità alle voste fatiche.

[00019889.exe]

JAVA

dBarCode Java Class DataMatrix

Per gestire codici a barre di forma quadrata e rettangolare da ruotare a piacere.

[00024654.EXE]

Tra le nuove funzioni introdotte con la versione 3.0:

- Stampa dei diagrammi
- Import di sorgenti
- Possibilità di *un-do*
- Copy verso la clipboard di Windows
- Print preview a pagine multiple

Gratuito, i file di export e stampe riportano la scritta "not for commercial use".
PoseidonCE_3_0_0_Installer.exe

Softsilver Transformer 2.5.2

Riversare dati da ODBC in XML

Un tool che può convertire in XML sia file di testo sia dati da fonti ODBC, senza alcun intervento da parte dello sviluppatore. È possibile fissare i parametri attraverso cui Softsilver interpreterà i file di testo e si possono specificare le query SQL con cui verranno estratti i dati dal database. L'impostazione del file XML di output è possibile definirla attraverso delle semplici operazioni di drag&drop, mentre l'operazione di conversione può avvenire attraverso una semplice utility a riga di comando. La versione 2.4.2 presenta nuove e più flessibili opzioni nella formattazione dell'output, oltre a migliorare il supporto per i template. Versione di valutazione valida trenta giorni.
st25setup.msi

PureBasic 3.92

BASIC… a tempo di record!

Un linguaggio basato sulle regole del BASIC, ma con notevoli miglioramenti. La possibilità di compilare il codice prodotto rende PureBasic una scelta interessante per i neofiti così come per i più esperti che hanno bisogno di qualche programmino di rapido utilizzo. La proverbiale semplicità del BASIC si coniuga in questo caso con buone prestazione ed ottime capacità di gestione della grafica. Installatelo e date un'occhiata a qualcuno dei numerosi esempi presenti: sarete stupiti del rapporto qualità/righe di codice! A dispetto di una sintassi veramente elementare, PureBasic mette a disposizione degli sviluppatori strumenti come puntatori, strutture, procedure e liste dinamiche.

Non un semplice strumento per principianti, ma un interessante ambiente di svi-

luppo che fa della semplicità il suo punto di forza. Questa versione dimostrativa è limitata a compilare solo 800 linee di codice e risulta inibito l'accesso alle API Win32.
PureBasic_Demo.exe

SuperEdi 3.5

Un editor piccolo e funzionale per gli sviluppatori

Ideato appositamente per i programmatori, SuperEdi può essere utilizzato sia per lo sviluppo in locale che per modificare file in remoto. Completamente gratuito, presenta tutte le principali funzionalità degli editor più blasonati: evidenziazione sintattica per i maggiori linguaggi, filtri per la manipolazione automatica del testo e supporto multilingua. Gli utenti possono estendere le potenzialità di SuperEdi attraverso script scritti in JScript, VBScript o Perl: tutte le funzioni di SuperEdi sono presenti sotto forma di script in una cartella all'interno della directory di installazione di SuperEdi. È sufficiente aggiungere nuovi script o modificare quelli già presenti per plasmare SuperEdi a nostro piacimento. Gratuito, da provare!

SuperEdi-3.5.U.exe

ColorCache 3.0

Un aiuto nella scelta dei colori

Un tool completo e semplice da utilizzare che ci aiuta nel comporre l'interfaccia delle nostre applicazioni. Possiamo individuare (con un "contagocce") il colore presente in una zona qualsiasi dello schermo e, a partire da un qualsiasi colore, possiamo farci indicare quali sono i complementari, i più simili, quelli che maggiormente contrastano e così via. Le palette create sono in XML e possono essere esportate verso numerosi formati: ACT, .ACO, .PAL, .AI, e .ACF . Versione di prova valida quindici giorni.
cche3000.exe

JFrameBuilder 3.0.1

Per creare sofisticate interface

Un tool che semplifica la realizzazione di interfacce grafiche per applicazioni Java Swing. Attraverso un approccio drag& drop, si risparmia all'utente la maggior parte del lavoro di codifica.
Il codice generato è strutturato in modo da essere perfettamente intellegibile per i programmatori. Non richiede alcuna libreria aggiuntiva e fornisce il supporto per tutti i principali settagli. Versione di valutazione dimostrativa, consente di integrare un massimo di quindici componenti per progetto.
JFB_301.zip

ASP.NET Maker 1.1

Crea codice automaticamente

Un generatore automatico di codice che, a partire da un base di dati, può creare intere applicazioni ASP.NET. I database supportati sono Access, SQL Server, Oracle e qualsiasi DB pilotabile via ODBC o ADO. Sono già disponibili alcune funzioni standard built-in come la rappresentazione master-detail, la sicurezza sulle transazioni, l'aggregazione sui campi e altro ancora. L'utente può scegliere il linguaggio in cui verrà prodotto il codice, fra VB.NET e C#. Versione di prova valida trenta giorni.

aspnetmkr.exe

Quick License Manager 2.1

Proteggi il tuo software dai pirati

Un sistema di gestione delle licenze molto semplice ed efficace che si occupa di creare chiavi di attivazioni sicure per proteggere il software che sviluppiamo dalla pirateria. Le chiavi di attivazioni si creano con un clic mentre, per il software, è disponibile un'apposita API COM a invocare per la validazione delle chiavi stesse. In meno di un'ora potrai trasformare il tuo software in una trial! Versione di prova valida quindici giorni.

LIBRERIE JAVA

WebCab Probabilità e Statistica

La classe giusta per le funzioni di probabilità e statistica che vi servivano.
[00024384.EXE]

ElegantJ Printer

L'aiuto che vi serve per gestire la stampa dalle vostre applicazioni in Java.
[00022604.EXE]

PDFlib

Come generare velocemente un PDF senza sapere come è fatto un PDF.
[00022834.EXE]

LIBRERIE PERL

RSS Maker

Come realizzare e mantenere file nel formato RSS.
[makerss.zip]

Moniker

Uno script a linea di comando di facile installazione.
[moniker-0.2.zip]

RSA Encryption in Perl

Come crittografare un messaggio utilizzando l'algoritmo RSA.
[Tutorial16.zip]

Un Proxy in Perl

Miglioriamo la navigazione bloccando i fastidiosi popup!
[ProxyPerl.zip]

Verifica di un indirizzo e-mail

Come verificare l'esistenza degli indirizzi e-mail contenuti in un file.
[VerificaMail.zip]

LIBRERIE VBNET

ChilKat MTH

Automatizzare la conversione e l'invio di pagine HTML via e-mail.
[ChilkatMht.exe]

ChilKat MTH

Automatizzare la conversione e l'invio di pagine HTML via e-mail.
[ChilkatMht.exe]

LIBRERIE VB.NET

DevMail .NET

Potenza e semplicità d'uso nella gestione delle e-mail.

[devMail.Net.msi]

Folder Browser Control

Un esempio per una shell in piena regola.

[FolderBrowser.msi]

Grid View Component

Da una ListView ad un Grid Control attraverso un User Control.

[GridViewRTMCode.zip]

ComponentOne Chart per .NET

Grafici in due e tre dimensioni, generati in modo veloce e altamente professionale.

[00021848.EXE]

LIBRERIE PHYTON

Fraction

Una classe per le frazioni in Phyton.

[fraction.tar.gz]

Matfunctions

Matrici, vettori e tutte le routine necessarie alla loro manipolazione.

[Matfunc.py.zip]

Statistics

Anche per Phyton una raccolta di funzioni statistiche di base.

[Stats.py.zip]

qlm.exe

DotNet2UML 2.1

Legge gli assembly prodotti da .NET e crea una rappresentazione UML portabile

Un piccolo tool che può rivelarsi di grande utilità nelle operazioni di reverse engineering: DotNet2UML può leggere un qualsiasi assembly .NET (IL) e riportarlo in formato UML, consentendone così la comprensione. L'output fornito da DotNet2UML è in XMI (XML Metadata Interchange), un formato accettato da pressoché tutti gli ambienti di progettazione UML. Gratuito.

AssemblyMetadata2XMI.zip

XML:Wrench 1.2.1

Manipolazione di file XML

Un editor XML decisamente spartano che fa della semplicità il suo punto di forza. Con XML:Wrench è possibile manipolare anche file HTML e XHTML, il tutto senza inutili fronzoli e con un'interfaccia che piacerà agli utenti più smaliziati e meno avvezzi alle funzionalità che rallentano la macchina prima ancora di velocizzare il lavoro dello sviluppatore!

Sono comunque disponibili alcune indispensabili funzionalità come l'auto-completion, e la possibilità di salvare i setting in base al progetto. La migliore caratteristica: è gratuito.

xmlwrench-v121.exe

Notepad++ 2.6

Un super notepad!

Con l'interfaccia in Italiano, un editor gratuito (distribuito con licenza GPL) orientato alla manipolazione di codice e che offre il supporto a tutti i più diffusi linguaggi di programmazione: C, C++, Java, C#, XML, HTML, PHP, Javascript, file .ini, file batch, ASP, VB/VBS, SQL, CSS, Perl, Python, Fortran, actionscript e molti altri ancora.

Il sintax highlighting è ottimamente realizzato e, attraverso le funzionalità di drag & drop, sarà semplicissimo utilizzare Notepad++ anche in progetti già avviati. Provatelo: l'editor è davvero velocissimo e, lavorare con su documenti XML, anche molto lunghi, diventa un vero piacere. Ampiamente personalizzabile: per i programmatori che non devono chiedere mai!

npp.2.6.Installer.exe

JetBrains ReSharper 1.0.4

Un assistente per C# in Visual Studio

Un add-in per Visual Studio .NET 2003 che fornisce un aiuto "intelligente" durante la scrittura di codice C#. Oltre a fornire il rilevamento in tempo reale degli errori di battitura, ReSharper fornisce anche la soluzione più corretta (o più probabile) al nostro problema.

Molto interessanti anche le funzionalità di refactoring offerte dall'addi-in. Versione di prova valida trenta giorni.

ReSharper1.0.4.exe

TOOLBAR CREATOR: LA NOSTRA PRIMA BARRA

1 **Avviamo l'applicazione e specifichiamo nome e cartella del nostro progetto. Un clic su *Create Toolbar*, e la struttura dell'applicazione sarà pronta realizzata.**

2 **In Visual Studio andiamo a selezionare il progetto appena creato. Il progetto è facilmente modificabile in tutte le sue parti: il template chiarisce ogni dubbio.**

3 **Ed ecco come si presenta la nostra prima barra, come vedete abbiamo c'è un piccolo logo che spero riconoscerete! Tutti i pulsanti sono già funzionanti.**

TOMCAT 50

L'application server per le vostre applicazioni JSP

Tomcat è il tool fondamentale per iniziare a creare applicazioni JSP. Diciamo subito che come categoria è un Web Server, al cui interno possono girare applicazioni scritte in JSP, ovvero l'analogo delle famosissime pagine ASP ma in JAVA.

Directory /tomcat

INSTALLAZIONE

Dopo avere effettuato l'installazione di Tomcat, tramite il comodo installer integrato, vi ritroverete a disposizione un Web Server sulla porta 8080. Provate ad accedere a http://localhost:8080 e vi risponderà la classica pagina d'avvio di Tomcat.

INTEGRARE TOMCAT IN APACHE

Prima di tutto avete bisogno del "connector adeguato". I file da considerare sono

- *mod_jk_1.2.6_1.3.31* per Apache 1.3.x.
- *mod_jk_1.2.6_1.3.31_EAPI* per Apache 1.3.x con il supporto SSL
- *mod_jk_apache_1.2.6_2.0.50.dll* per Apache 2.0.50
- *isapi_redirect_1.2.6.dll* per IIS 5.

Scegliete il connector adeguato, rinominatelo *mod_jk.dll*, e copiatelo all'interno della directory *modules* nella home di Apache. Dalla directory conf della home di Apache editate *HTTPD.CONF* e aggiungete le seguenti righe:

```
LoadModule jk_module modules/mod_jk.dll
AddModule mod_jk.c
JkWorkersFile "C:\Programmi\Apache
        Software Foundation\Tomcat 5.5\conf\
                    workers.properties"
JkLogFile "C:\Programmi\Apache Software
        Foundation\Tomcat 5.5\logs\jk_log.txt"
JkLogLevel debug
JkLogStampFormat "[%a %b %d
                    %H:%M:%S %Y] "
JkMount /*.jsp ajp13
JkMount /jsp-examples/* ajp13
```

Con queste abbiamo detto ad apache che tutte le richieste a pagine JSP devono essere elaboratoe dal connettore *ajp13* di Tomcat e che tutte le pagine richiamate dopo */jsp-examples* devono essere elaborate da Tomcat. Ci sarebbe qualche altra cosa da dire in merito a questo connettore e alla sua configurazione, così come si dovrebbe dire qualcosa sui file *workers .properties* e *server.xml* presenti nella directory di configurazione di Tomcat. Ma in Tomcat 5.5 questi due file sono gia configurati correttamente perciò tutto funzionerà correttamente. A questo punto non ci resta che riavviare Apache e puntare il browser verso *http://localhost/jsp-examples* per vedere cosa si può fare con le JSP.

LA NOSTRA PRIMA PAGINA JSP

In una qualunque directory create un file index.jsp come segue

```
<html>
<body>
<%
   out.println("Hello ioProgrammo World");
%>

</body>
</html>
```

In una sotto directory *WEB-INF* relativa alla root di *index.jsp* create un file *web.xml* come segue:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun
Microsystems, Inc.//DTD Web Application 2.3
        //EN" "http://java.sun.com/dtd/
                web-app_2_ 3.dtd">
<web-app><servlet>
   <servlet-name>HelloWorldExample
                </servlet-name>
   <jsp-file>/jeiessepi/index.jsp</jsp-file>
 </servlet>
</web-app>
```

A questo punto usate il comando *jar* per creare un file *war* della directory in questione:

```
jar -cvf ../iop.war *
```

infine puntate il browser su http://localhost:8080 cliccate su 'Tomcat Manager', utilizzate il text box "select war to deploy" e inserite il file war creato in precedenza.
La vostra applicazione TomCat sarà così creata. Naturalmente questo processo è necessario solo per creae l'applicazione, le successive modifiche ai file JSP non richiedono di rifare tutto questo processo.
Per vedere il risultato della vostra pagina puntate su http://localhost/iop/index.jsp

ToolbarCreator 1.0

Crea la tua toolbar per Internet Explorer

ToolbarCreator fornisce un workspace (in Visual Studio 6 e Visual Studio .NET) per la realizzazione di toolbar per Internet Explorer. Per avere un'idea delle possibilità che si aprono con ToolbarCreator, possiamo dire che è possibile realizzare un barra sullo stile di Google toolbar in poco meno di trenta righe di codice. Tra i componenti a nostra disposizione: pulsanti, toggle button, menu, etichette, combo-box, controlli editabili e controlli di sola lettura.

ToolbarCreator10Setup.exe

LIBRERIE PHYTON

Genaut

Enigmistica e programmazione si incontrano sul pianeta Phyton.

[Genout.py.zip]

Battle

Simulazione di space battle interessante dal punto di vista della programmazione.

[battle.zip]

Trasformazione di stringhe con tecniche di Lindermayer

La grafica dei sistemi caotici

Affronteremo la rappresentazione grafica di sistemi caotici, come quelli di Lindermayer. Realizzeremo una piccola applicazione che, nella sua semplicità, consentirà la visualizzazione dei primi frattali

Utilizza questo spazio per le tue annotazioni

Analizzare il caos, con un approccio che ne sveli le dinamiche, al fine di sviluppare applicazioni software che lo simulino è stato un utile esercizio. Ancora una riprova che l'evoluzione di molti fenomeni naturali e artificiali non sempre segue logiche che tendono verso bassi livelli di energia. Insomma, non sempre sistemi aggregati di elementi sono portati a organizzarsi verso forme più stabili e semplici. In poche parole si è intuito che l'entropia per molti sistemi è in continuo aumento. Dopo aver osservato le leggi con le quali un particolare sistema caotico (il sistema di Lindermayer) si sviluppa ci occupiamo adesso di come esso può essere rappresentato graficamente. L'obiettivo è trovare una relazione tra le produzioni tipiche di tali sistemi, che come abbiamo visto sono delle stringhe, e le relative rappresentazioni grafiche. Ovviamente, si tratta di una delle numerosissime possibilità di realizzazione grafica. Prima di cominciare, invito anche chi non ha letto gli articoli precedenti della mini serie a proseguire: i lettori sapranno che, a rischio di essere leggermente ripetitivi, facciamo un grande sforzo per rendere ogni articolo sufficientemente autonomo. Ovviamente resta auspicabile una lettura dell'intera mini serie per avere una visione globale e più ampia.

TANTO PER COMINCIARE

Il risultato della puntata precedente è stato la costruzione di un programma per l'implementazione di un sistema di Lindermayer. Certo abbiamo anche esplorato in generale sistemi caotici, visionando a più livelli di approfondimento le diverse tecniche di studio. Abbiamo, però, concentrato la nostra attenzione al semplice processo iterativo di associazione di una stringa iniziale (assioma) ad una risultato (produzione) ottenuta applicando semplici regole. Così, abbiamo potuto osservare come la stringa cresce, applicando nei vari stadi (generazioni) sempre le stesse regole di produzione. Si è notato come tale processo sia in grado, con i dovuti livelli di approssimazione, di simulare molti fenomeni di crescita. Il metodo è infatti applicabile a flora, popolazioni di esseri viventi (dalle formiche all'uomo) e a molti altri casi. Il nostro lavoro si è limitato all'applicazione e successiva osservazione della "crescita" di stringhe. Naturalmente, l'interpretazione che si può dare alla sequenza di simboli dipende dall'ambito a cui si associa l'esperienza. Nel presente articolo daremo un significato meramente grafico che ci aiuterà a costruire forme autosimilari; un ulteriore modo per produrre i frattali. Di seguito è riportato un assioma (stringa di caratteri) e una produzione (costituita da una sola regola).

Assioma: *A+A+A+A*
Regola di produzione: *A -> A+A-A-AA+A+A-A*

Il sistema è alquanto semplice, sebbene caotico. È costituito da una stringa assioma con due soli simboli (*A*,+) ripetuti; e da una sola regola di produzione associata al simbolo *A*. Il risultato della prima generazione è riportato di seguito.

A+AA-A+A+A+AA-A+AA+AA-A+A-A+AA-A+A+A+AA-A+A+A+AA-A+A+A+AA-A+AA+AA-A+A-A+AA-A+A+A+AA-A+A

Il programma *lsyst3.pas* prodotto nello scorso appuntamento, riesce a gestire questa fase.

DALLA STRINGA ALLA GRAFICA

Seguendo l'esempio appena descritto, passiamo

all'interpretazione grafica. Vedremo che la generalizzazione è agevole. Si tratta di associare ad ogni simbolo un significato grafico. Il sistema prodotto, in definitiva, consta di soli tre simboli: *A*, +, -. Una possibile interpretazione grafica è:

- **A:** traccia un segmento di lunghezza prefissata nella direzione corrente;
- **+:** svolta a destra di un angolo prefissato;
- **-:** svolta a sinistra di un angolo prefissato.

Il risultato grafico si potrà apprezzare esaminando l'output del programma che stiamo per costruire. Prima di procedere allo sviluppo del programma sono necessarie alcune indispensabili puntualizzazioni geometriche. La rappresentazione è su un sistema di assi cartesiani bidimensionale. In ogni istante si deve disporre di un punto corrente; e spesso bisogna calcolare un nuovo punto. Come mostrato in **Figura 1**, si può notare che per produrre un nuovo punto bisogna tenere conto di un angolo di rotazione q (o *theta*), e del senso di rotazione. In particolare sono stati prodotti due nuovi punti per una rotazione oraria e una antioraria di un angolo pari a 90°. L'angolo g (o *gamma*) indicherà la direzione del punto e qualora si debba cambiare direzione, (dell'angolo *+q* in corrispondenza di + e *-q* per il simbolo -) esso sarà aggiornato. Il segmento verrà tracciato tra i due punti a disposizione. Il primo esempio che stiamo costruendo fa riferimento a un angolo theta fisso di 90°. In generale, in corrispondenza di un punto *p* con le sue coordinate *p.x* e *p.y* il nuovo punto *np* dopo uno spostamento orario *q*, si ottiene dalla semplice applicazione di una regola trigonometrica:

$$np.x=p.x+k*cos(g+q)$$
$$np.y=p.y-k*sen(g+q)$$

La rotazione antioraria è data dalla formula precedente che cambia per i soli angoli.

CODICE SOTTO LA LENTE DI INGRANDIMENTO

Nel codificare la soluzione faremo ovviamente riferimento al risultato ottenuto il mese scorso. Si tratterà quindi di aggiornare e ampliare alcune parti e di aggiungerne di nuove. Ad esempio, la fase preliminare e fondamentale di dichiarazione di tipi e variabili è stata profondamente estesa.

```
uses graph;
const
    (*massimo numero di regole e generazioni*)
```

```
    gmax=10;
    (*Coordinate grafiche del punto di partenza *)
     xcentr=400;
    ycentr=300;
(* I tipi definiti dall'utente:
  - lista a puntatori (tpunt);
  - vettore lista a puntatori (tvetpunt)
  - punto sul piano cartesiano, coppia di coordinate;
  - tipo di rotazione, enumerato (rotazione)
*)
type  tpunt=^nodo;
    nodo=record
        c:char;
      next:tpunt;
      end;
    tvetpunt=array[1..gmax] of tpunt;
    punto=record
         x,y:integer;
        end;
    rotazione=(orario,antiorario,invariato);
var   (* Assiomi (s) e regole di produzione (pr) *)
     pr,s : tvetpunt;
     (* numero di generazioni (n),
       generazione corrente (g) *)
     n,g : integer;
     (* costante grafica e fattore di riduzione*)
     k,fr:real;
     (* Angolo di rotazione *)
     theta:real;
```

Si trovano nuovi elementi, tra costanti, tipi e variabili. Analizziamoli con la lente del programmatore. Le due costanti *xcentr* e *ycentr* indicano le coordinate di schermo del punto di partenza nella rappresentazione grafica. Ricordo che, per semplicità e chiarezza, ho deciso di continuare a produrre il codice in Pascal (si tratta della versione free di bloodshed: *www.bloodshed .net*) con riferimento alla tipica piattaforma grafica descritta dalla conosciuta libreria *graph*. A tale proposito, penso che tale codifica sia facilmente riportabile su qualsiasi ambiente grafico: dai canvas di Java e C++ a piattaforme programmabili come *OpenGL*. Ai tipi per la rappresentazione delle stringhe di qualsiasi lunghezza, implementati mediante liste a puntatori, si aggiungono due semplici ma utili elementi. *punto* è un record costituito da due elementi, *x* e *y*, che indica proprio un punto sul piano; mentre *rotazione* servirà per tenere traccia di cambi di direzione (orario o antiorario), questa può anche non cambiare (invariato). Tra le variabili globali le nuove sono tre: *k*, *fr* e *theta*. La prima indica l'ampiezza del segmento (o altro elemento grafico) che si dovrà tracciare. Da notare che, per le caratteristiche di autosimilarità, tale valore dipende dalla generazione: per intenderci, la rappresentazione grafica della prima generazione avrà elementi più gran-

Fig. 1: ***Sistema bidimensionale di assi cartesiani. Interpretazione geometrica***

FRACTUS

Fractus è una parola latina che significa frastagliato. Il nome si deve al padre dei frattali: B. Mandelbrot, il quale, a quanto si narra, imbattendosi nel vocabolario di latino del figlio, studente di medicina, trovo la parola fractus che considerò appropriata per descrivere il suo studio.

di corrispondenti a un fissato valore di *k*. Nelle successive produzioni *k* verrà ridotto di un fattore (*fr*) letto in input. Così, gli elementi grafici saranno più piccoli e in numero maggiore. L'esempio riportato nell'ultimo paragrafo chiarisce il concetto. In tal modo si garantiscono le caratteristiche di autosimilarità. La variabile *theta* esprime l'angolo di rotazione, a rigore può non essere fissa. La novità è quindi la routine grafica chiamata *disegna* insieme alla funzione di servizio nuovacoo. Ecco il codice:

```
(* calcola il nuovo punto in funzione del cambio di
                                          direzione *)
function nuovacoo(psenso:rotazione; var ga:real;
              th:real; pcoo:punto; cost:real):punto;
var vlcoo:punto;
Begin
     if psenso=orario then ga:=ga + th
     else if psenso=antiorario then ga:=ga - th;
     vlcoo.x:=pcoo.x+round(cost*cos(ga));
     vlcoo.y:=pcoo.y+round(cost*sin(ga));
     nuovacoo:=vlcoo;
End;
(*disegna la produzione associando i simboli a
                               realizzazioni grafiche*)
procedure disegna (ns:tpunt; pk:real);
var  i,D,Gra : integer;
        cs : tpunt;
    coo,ncoo : punto;
      gamma : real;
      senso : rotazione;
Begin
     (*Inizializzazione grafica*)
     D:=DETECT;
     initgraph(D,Gra,'');
     setcolor(RED);
     (*puntatore di comodo posizionato sul parametro*)
     cs:=ns;
     (*configurazione di parametri:
       - angolo iniziale gamma
       - coordinate iniziali
     *)
     gamma:=0;
     coo.x:=xcentr;
     coo.y:=ycentr;
     (*punto di partenza*)
     circle(coo.x,coo.y,3);
     circle(coo.x,coo.y,4);
     ncoo:=nuovacoo(invariato,gamma,theta,coo,pk);
     while cs<>nil do
     begin
     senso:=invariato;
     case cs^.c of
     'A': line(coo.x,coo.y,ncoo.x,ncoo.y);
     'B': begin
          ncoo:=coo;
          circle(ncoo.x,ncoo.y,2);
          circle(ncoo.x,ncoo.y,4);
          end;
     'C': rectangle(coo.x,coo.y,ncoo.x,ncoo.y);
     'D': circle(ncoo.x,ncoo.y,2);
     'E': putpixel(ncoo.x,ncoo.y,YELLOW);
     '+': begin
           ncoo:=coo; (* si torna al punto precedente *)
           senso:=orario;
          end;
     '-': begin
           ncoo:=coo;
           senso:=antiorario;
          end;
     end;
     coo:=ncoo;
     ncoo:=nuovacoo(senso,gamma,theta,coo,pk);
     cs:=cs^.next
     end;
     readln;
     closegraph;
end;
```

Analizziamo dapprima *disegna*. Esaminiamo rapidamente le singole variabili. Con *gamma* si esprime l'angolo corrente in radianti. Grazie alle proprietà periodiche, anche se le rotazioni saranno sempre e solo in un senso, ad esempio quello antiorario (simbolo -), il programma funziona correttamente. *Ricordo che un angolo a produce la stessa rotazione in un piano dell'angolo a+360, in generale di a +q*360, con q qualsiasi tra gli interi*. I due punti *coo* e *ncoo* sono rispettivamente il *corrente* e il *nuovo*. Il puntatore *cs*, partendo dalla testa, scorrerà su tutta la stringa. Dopo la fase preliminare di configurazione delle variabili e delle modalità, si passa all'esame dei singoli simboli della stringa e della relativa associazione grafica. Il ciclo di *while* è predisposto al compito. A puro titolo di esempio, sono proposte 7 regole, ma questa sezione si può personalizzare a piacimento. Il simbolo *A* disegna un segmento dal corrente al nuovo punto. La seconda regola *B* traccia due cerchi concentrici. In sequenza, le successive descrivono un rettangolo, un piccolo cerchio ed un punto entrambi a seguito di uno spostamento. Le rotazioni sono effettuate in corrispondenza dei simboli + e -. Dopo aver riportato il nuovo punto a quello corrente (operazione necessaria poiché altre regole calcolano automaticamente il nuovo punto), si invoca la funzione *nuovacoo*. I parametri sono il senso (orario +, antiorario – e *invariato*), l'angolo globale *gamma* e quello relativo *theta*; nonché il punto corrente e la costante grafica. Tale funzione, applicando la legge trigonometrica di rotazione prima esposta provvede al calcolo di un nuovo punto, ossia il valore restituito. Se il senso è orario, l'angolo globale si aggiorna sommando quello relativo, se

SERIE DI NEWTON

Una particolare tecnica di descrizione dei sistemi caotici tiene conto delle radici ottenute dall'applicazione della serie di Newton.

f(z) =a0 + a1 z + a2 z2 + ... + am zm = 0

Una serie dove la n+1 esima approssimazione alla soluzione è data da:

zn+1 = zn –f(zn) / f'(zn)

con *f'(zn)* come pendenza (derivata) di *f(z)* valutata su *zn*. Si può creare un immagine 2D, in cui ogni punto si trova in una partizione del piano. Tale informazione può essere usata per colorare il punto di partenza *z0*, a seconda che soddisfi o meno alcune condizioni. Ragionamento analogo a quello fatto da *Julia* per la rappresentazione grafica dei suoi insiemi.

antiorario l'angolo relativo viene sottratto; infine, se rimane invariato, *gamma* non cambia. Ad ogni iterazione si scorre sulla lista *cs* per esaminare il nuovo simbolo e si aggiornano punti correnti *coo* e nuovi *ncoo*.
Ultimo passo è richiamare le nuove routine prodotte. Si tratta ancora di un aggiornamento del codice già prodotto.

```
Begin (* main *)
    writeln;
    writeln('-- Lindermayer System --');
    writeln(' # _ una realizzazione de ilmagodifibra #');
    writeln(' * implementazione: liste a puntatori *');
    writeln;
    config(s);
    carica(s[1],'Assioma');
    caricpr(pr);
    write('N. rig. --> ') ;
    readln(n);
    write('k iniziale --> ') ;
    readln(k);
    write('Fattore di riduzione --> ') ;
    readln(fr);
    write('Rotazione in gradi --> ') ;
    readln(theta);
    (*Trasformazione in radianti*)
    theta:=theta/57.3248;
    for g:=1 to n do
    Begin
      produzione(s[g+1],s[g],pr);
      writeln(g+1,' > ');
      output(s[g+1]);
      disegna(s[g+1],k);
      k:=k/fr;
      writeln;
    End;
End.
```

Dopo l'input di tutte le variabili globali e dell'assioma si passa al ciclo che produce le stringhe nelle diverse generazioni. In tale ciclo si richiama anche la rappresentazione grafica della stringa prodotta attraverso *disegna(s[g+1],k)*; inoltre, si aggiorna la costante k riducendola ad ogni generazione del fattore di riduzione fr.

ESEGUIAMO LA NOSTRA APPLICAZIONE

È il momento di gustare i risultati ottenuti. Applichiamo innanzitutto il programma all'esempio proposto in principio. In figura 2 è mostrata la sua rappresentazione nelle sue quattro fasi . La costante grafica k è posta a *72*, *fr* a *4* e *theta* a *90°*.
Le caratteristiche di autosimilarità risaltano immediatamente nella rappresentazione grafica. Inoltre, la riduzione del segmento (*k*) ad ogni generazione di un fattore 4 fa si che la dimensione finale della figura, intesa come ingombro, rimanga pressoché costante.
Nell'esempio, è stata sfruttata solo una piccola parte della potenzialità del programma, infatti, vi è una sola regola di produzione.
Un secondo esempio è stato prodotto applicando tre regole di produzione e cinque simboli. In **Figura 3** è riportato una videata dell'esecuzione del programma, dove si possono scorgere gli input e gli output (delle prime due generazioni). Sebbene la figura sia autoesplicativa sono da puntualizzare alcuni aspetti: l'angolo di variazione questa volta è 45°; la presenza delle nuove regole B e C; associate alla produzione di cerchi e rettangoli. La tripletta di + nella prima regola di produzione indica una rotazione di 135° (45*3). Anche in questo caso, l'assioma corrispondente alla traccia di un quadrato (per la presenza delle quattro rotazioni orarie, +) di quattro elementi grafici. In **Figura 3** sono riportate le prime tre generazioni.

Fig. 2: ***Un l-system nelle sue quattro generazioni***

Fig. 3: ***Esecuzione del programma***

CONCLUSIONI

Spero che l'esplorazione di questo ambito, ricco di possibili applicazioni nel campo reale, sia stato apprezzato. Certo ci sarebbe molto da esaminare e da implementare. Ci riserviamo questo privilegio per il futuro. Nel contempo può essere stimolante aggiungere nuove componenti al programma prodotto, introducendo ad esempio nuovi associazioni simbolo-effetto grafico. Altrettanto divertente è testare il programma con personali input. In questo ultimo caso l'obiettivo può essere creare apprezzabili effetti grafici.

Fabio Grimaldi

Fig. 3: ***Prime tre generazioni per l.system di figura 3***

Costruzioni matematiche sulla manipolazione di griglie di numeri

Soluzioni per quadrati magici

Individuare una soluzione per quadrati magici è un piacevole esercizio manuale in caso di piccole dimensioni. Per un considerevole numero di righe e colonne si impongono metodi algoritmici

Utilizza questo spazio per le tue annotazioni

Vediamo come è possibile costruire con un programma, una matrice quadrata che ha la "magica" caratteristica di avere invariante (sempre lo stesso numero) le somme dei suoi elementi per riga, colonna e per le due diagonali. Il metodo che ci apprestiamo a studiare si può applicare a quadrati la cui dimensione è un multiplo di quattro. Per le altre dimensioni esistono altre tecniche risolutive. Nel caso in esame deve persistere la semplice relazione:

```
n = 4k
```

dove n è la dimensione del quadrato e k è una costante che vale un quarto di n. Essendo n un multipli di quattro, entrambe le grandezze: n e k sono dei numeri interi. Si parte considerando un quadrato che contenga i primi n2 numeri disposti per riga.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Il quadrato viene partizionato in nove parti ognuna delle quali è identificata con delle lettere, così come indicato dalla **Figura 1**.

Fig. 1: ***Partizionamento del quadrato in nove sotto aree***

Il partizionamento produce nove aree non tutte quadrate e comunque di dimensioni diverse, in cui ricorre la dimensione k oppure 2k, come mostrato in figura. Adottando una colorazione tipica delle scacchiere, con una alternanza scuro chiaro, distinguiamo due parti.

Il metodo proposto consiste nello scambiare le parti chiare: B con H e D con F. Lo scambio dei blocchi avviene in modo speculare; il primo elemento del blocco, nella nuova posizione diventerà l'ultimo e viceversa.

Ma vediamo esempi per chiarire il procedimento. Le dimensioni n=4 e n=8 sono entrambe utili. Ecco i quadrati magici che producono.

1	15	14	4
12	6	7	9
8	10	11	5
13	3	2	16

1	2	62	61	60	59	7	8
9	10	54	53	52	51	15	16
48	47	19	20	21	22	42	41
40	39	27	28	29	30	34	33
32	31	35	36	37	38	26	25
24	23	43	44	45	46	18	17
49	50	14	13	12	11	55	56
57	58	6	5	4	3	63	64

Si può immediatamente verificare come si tratti effettivamente di quadrati magici. Realizzare un programma per computer sul metodo risulta semplice.

Ecco un frammento di codice C++ che si occupa della parte cruciale della questione.

```
void scambiablocchi()
{
  // Commutazione dei blocchi B e H di dimensione (k*2k)
  // caricamento del blocco B sulla matrice temp1
  for (i=0; i<k; i++)
    for (j=0; j<2*k, j++)
      temp1[i][j]=quad[i+k][j+k];
  // caricamento del blocco H al posto del blocco B
```

```
for (i=0; i<k; i++)
  for (j=0; j<2*k, j++)
     quad[k-i+1,3*k-j+1]=quad[i+3*k,j+k];
// caricamento del blocco H (temp1) al posto del
                                             blocco B
for (i=0; i<k; i++)
  for (j=0; j<2*k, j++)
     quad[4*k-i+1, 3*k-j+1]<-temp1[i,j];
… }
```

I puntini indicano che lo scambio deve essere realizzato anche per i blocchi D e F.
La matrice quad è il risultato sperato, il quadrato magico. Un elemento degno di nota è la permutazione delle matrici in ordine inverso.

A tale proposito si osservino come sono usati gli indici i e j con riferimento alle sottomatrici di quad e alla matrice temporanea temp1.
La procedura sarà richiamata previo il caricamento della matrice con i primi n2 numeri naturali; e il controllo che n sia effettivamente un multiplo di quattro.

CONCLUSIONI

Quello esaminato è soltanto uno dei metodi automatici. Esistono altre tecniche che avremo modo di osservare in futuro.

Fabio Grimaldi

L'ANGOLO DELLA COMPETIZIONE

Alimentiamo ancora lo spazio dove misuriamo il nostro QI. Propongo inizialmente due nuovi problemi e di seguito do la soluzione ai passati.

Q Si consideri una scacchiera di cinque caselle di lato. Qual è il massimo numero di regine posizionabili nella scacchiera senza che si tengano reciprocamente sotto scacco?
Risposta aperta

Fig. 2: Osserviamo come si diffonde il contagio (b) a partire da una situazione di iniziale infezione (a)"

Le regole di contagio mostrano come si infettino altre caselle al primo passo (indicate in figura 2-a con il numero 1) e al passo successivo (indicate con 2). Agli stadi ulteriori non si sviluppa il contagio che resta così circoscritto. Da alcune osservazioni si deduce che la totale diffusione del contagio in ogni casella delle righe e delle colonne avviene se almeno una casella è infetta per ogni riga e colonna. Il numero minimo è quindi 6. Una configurazione plausibile iniziale è riportata in figura 3-a.

Fig. 3: Osserviamo come si diffonde il contagio si diffonde nella totalità dell'area (c) a partire da una situazione di iniziale infezione (a) e una evoluzione al primo stadio (b)"

Nelle Figure 3-b e c si osserva prima l'iniziale stadio di contagio e poi tutti i successivi fino alla totale infezione.

SOLUZIONI ARTICOLI PRECEDENTI

Ecco adesso i problemi proposti nello scorso appuntamento e le relative soluzioni.

1 Si consideri un torneo di calcetto in cui ogni squadra deve incontrare esattamente una volte tutte le altre. Se il numero di partite è 136, quale è il numero delle squadre?
Il mio approccio a questo tipo di problemi è per così dire anglosassone o buttom-up. Ossia, nell'ordine svolgo le seguenti fasi: analizzo un caso semplice e lo risolvo, tendo di generalizzare la soluzione attraverso una formula o un algoritmo; applico il risultato appena ottenuto alla situazione richiesta. Nel caso specifico: analizziamo dapprima il semplice caso di un torneo con sei squadre. Indichiamo con le prime sei lettere dell'alfabeto le varie compagini. La squadra A dovrà fare cinque partite come riportato di seguito:

A-B; A-C; A-D; A-E; A-F;

Anche la squadra B dovrà fare cinque partite, come tutte le altre del resto; ma nella computazione del numero totale di partite non dovremo conteggiare l'incontro A-B, poiché è stato già fatto in precedenza. Si terrà conto di quattro partite:

B-C; B-D; B-E; B-F

Per C si dovrà tener conto di tre partite, per D di due, per E di una, e per F di nessuna. Nel semplice caso in esame si hanno quindi un numero di partite pari alla somma 5+4+3+2+1=15. Possiamo facilmente generalizzare a n squadre. Il numero di partite sarà la somma tra la successione di numeri consecutivi che va da 1 a n-1. Tale somma è anche molto facile da calcolare, anche senza il meccanico uso di una calcolatrice, sulla base della serie di Gauss (chi non ricorda l'aneddoto riguardante l'enfant prodige Gauss, che alle scuole elementari fece la somma dei primi cento numeri sulla base della semplice quanto geniale considerazione: il numero 101 si ripete cinquanta volte, come somma tra 1 e 100, 2 e 99, 3 e 98 e così via). Nel nostro esempio la formula che si ottiene è (n*(n-1))/2 che deve eguagliare 136 (il numero di partite). Il risultato, che si può trovare per tentativi, o matematicamente estraendo la n dalla equazione, è 17.

2 Si consideri un ipotetico gioco di realtà simulata che si effettua su un campo quadrato diviso in 6 righe e 6 colonne (contenente quindi un numero di caselle uguale a 36). All'inizio del gioco alcune caselle possono essere focolari di epidemie. Le epidemie si diffondono secondo il seguente schema: una casella viene infettata quando è adiacente ad almeno due delle caselle infette (due caselle sono considerate adiacenti quando condividono un lato; per esempio, due caselle vicine in diagonale non sono adiacenti in quanto condividono un solo punto). Quale è il numero minimo di caselle focolari di epidemia capaci di infamare tutto il campo da gioco?
Facciamo anche in questo caso un esempio per comprendere di cosa si tratta. Inventiamoci una configurazione con alcune caselle infette e verifichiamo cosa succede. Il primo esempio è riportato in Figura 2 a.

Biblioteca

INTRODUZIONE A MICROSOFT ASP.NET 2.0

Un buon libro per anticipare i tempi! Basato sulla Beta 1 di ASP.NET 2.0 (attesa per la primavera di quest'anno), il testo compre la quasi totalità delle funzioni che saranno disponibili nella versione finale. Ideato come introduzione alla piattaforma, il libro può diventare una vera guida alla programmazione, grazie ai numerosi esempi (oltre 70) e al taglio molto pratico tipico dell'autore. Il libro segue una divisione in quattro parti:

- Presentazione dell'ambiente Visual Studio 2005 e studio della classe Page, delle pagine master e delle Web Parts
- Accesso ai dati con tutte le novità di ADO.NET 2.0, il databinding, i nuovi componenti data source con i relativi controlli
- Introduzione ai nuovi servizi per le applicazioni: la possibilità di creare Wizard per il Web, generazione dinamica delle immagini, la gestione delle sessioni, il meccanismo di caching ed un capitolo intero dedicato alla sicurezza.
- L'ultima parte si concentra sugli argomenti più avanzati: a partire dalla traduzione delle vecchie applicazioni ASP.NET 1.1 per arrivare alle politiche di configurazione e monitoraggio dei server

Il libro viene venduto privo di CD ma tutti gli esempi sono disponibili on-line e liberamente scaricabili. Gli esempi sono tutti in C# e rappresentano un ottimo punto di partenza per un programmatore che voglia realizzare applicazioni Web personalizzate.

Difficoltà: Medio-Alta • Autore: Dino Esposito • Editore: Mondadori Informatica
http://education.mondadori.it/ • ISBN: 88-04-53736-1 • Anno di pubblicazione: 2004
Lingua: Italiano • Pagine: 432 • Prezzo: € 35,00

RETI WIRELESS – NOZIONI DI BASE

La diffusione delle reti wirless è in continuo aumento e, se per motivi professionali o per hobby, ci viene richiesto di installarli per conto di committenti o di amici, ma la nostra preparazione sull'argomento non è proprio elevata… ecco che un testo come quello presentato può fare la differenza. Con un linguaggio chiaro e semplice, il lettore troverà una panoramica sui concetti fondamentali e le tecnlogie alla base della comunicazione wireless, con informazioni pratiche e vari esempi di implementazione, sempre basati su casi reali. Senza ricorrere a tecnicismi, il testo insegna a implementare delle WPAN (wireless personal area network), WLAN (wireless local area network), MAN (metropolitan area network) wireless e WWAN (wireless wide area network). La sezione conclusiva analizza le minacce cui sono sottoposte le reti senza fili e le soluzioni per combatterle.
Un testo che non dà nulla per scontato e, anche grazie ad un efficace utilizzo delle immagini, consente di affrontare "da zero" il funzionamento delle reti senza filo. Sono affrontate tutte le più attuali tecnologie: dalle reti di cellulari al WiFi. Con un metodo "molto americano", alla fine di ogni capitolo è presente un piccolo test che permette di verificare la comprensione del testo da parte del lettore.

Difficoltà: Medio-Bassa • Autore: Jim Geier • Editore: Mondadori Informatica
Cisco Press www.ciscopress.com ISBN: 88-04-53928-3 • Anno di pubblicazione: 2004
Lingua: Italiano • Pagine: 238 • Prezzo: € 25,00

ON LINE

IT.COMP.LANG .VISUAL-BASIC

La maggioranza di voi avrà conoscenza di cosa sia un NewsGroups. Per quelli che non conoscono i newsgroups si sappia che si tratta di una sorta di forum di discussione organizzato in una modalità simile a quella del normale scambio di Email. Per raggiungere il newsgroups it.comp.lang.visual-basic potete settare l'apposito account in outlook.
All'interno del gruppo di discussione trovate moltissime persone disposte a darvi una mano nel risolvere i vostri problemi. Prima di tutto pero' consultate le FAQ per controllare se il vostro problema è stato risolto e affrontato in precedenza.
Vi si trovano una serie di consigli, trucchi, software, documentazione utilissima per i programmatori Visual Basic.
http://www.it-lang-vb.net

IL SITO DEL PITONE

Pyhton è sicuramente li linguaggio di scripting più interessante ed emergente dell'intero panorama della programmazione attuale. Il punto di partenza per chi volesse imparare a programmare in Python è ricchissimo di informazione e documentazione, raccoglie moltissimi link anche a risorse esterne oltre a tutorial e faq autoprodotti o tradotti da documentazione in inglese. ioProgrammo sicuramente

sosterrà Python nei prossimi numeri con articoli approfonditi, iniziare dunque a comprendere le basi del linguaggio è un buon punto di partenza.
http://www.python.it/